Technical Report 1113

Visual Navigation: Constructing and Utilizing Simple Maps of an Indoor Environment

Karen Beth Sarachik

MIT Artificial Intelligence Laboratory

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AI-TR 1113

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): Visual Navigation: Constructing and Utilizing Simple Maps of an Indoor Environment

5. TYPE OF REPORT & PERIOD COVERED: technical report

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): Karen Beth Sarachik

8. CONTRACT OR GRANT NUMBER(s): N00014-86-K-0685, N00014-85-K-0124

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS:
Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

12. REPORT DATE: March 1989

13. NUMBER OF PAGES: 91

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office):
Office of Naval Research
Information Systems
Arlington, VA 22217

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report):

Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

map making
mobile robot
motion vision
robot navigation
stereo vision

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See reverse

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Much work with mobile robots has been done in the past using both vision and sonar to build maps, or, given a map, to successfully plan and execute trajectories to a goal. The most successful examples of robot navigation occurred in carefully engineered environments where the robot was able to accurately predict what its sensory input should be at any point, and correct for drift by comparing actual input to the projected input. In unstructured environments however, the problem became much harder, and the obvious approaches failed to produce good results. The problem is further complicated by the fact that most interesting environments are not static, but rather are changing continually.

In this thesis I have attempted to attack the problem from a different angle altogether, using the way people navigate through buildings as insight and inspiration. The goal is to navigate through an office environment using only visual information gathered from four cameras, whose initial detailed configuration is not known, placed onboard a mobile robot. The method is insensitive to physical changes within the room it is inspecting, such as moving objects. The map is built without the use of odometry or trajectory integration, which are often unreliable. At the heart of this technique is the development of a "room recognizer" which is able to deduce the size and shape of a room in conjunction with a "door recognizer" which recognizes a potential door by finding two vertical edges close enough together. The long term goal of the project described here is for the robot to build simple maps of its environment, presumed to be a single floor of an office building, and to localize itself within this framework.

Visual Navigation: Constructing and Utilizing Simple Maps of an Indoor Environment

by
Karen Beth Sarachik

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

at the
Massachusetts Institute of Technology

March, 1989

Visual Navigation: Constructing and Utilizing Simple Maps of an Indoor Environment

by

Karen Beth Sarachik

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Abstract

Much work with mobile robots has been done in the past using both vision and sonar to build maps, or, given a map, to successfully plan and execute trajectories to a goal. The most successful examples of robot navigation occurred in carefully engineered environments where the robot was able to accurately predict what its sensory input should be at any point, and correct for drift by comparing actual input to the projected input. In unstructured environments however, the problem became much harder, and the obvious approaches failed to produce good results. The problem is further complicated by the fact that most interesting environments are not static, but rather are changing continually.

In this thesis I have attempted to attack the problem from a different angle altogether, using the way people navigate through buildings as insight and inspiration. The goal is to navigate through an office environment using only visual information gathered from four cameras, whose initial detailed configuration is not known, placed onboard a mobile robot. The method is insensitive to physical changes within the room it is inspecting, such as moving objects. The map is built without the use of odometry or trajectory integration, which are often unreliable. At the heart of this technique is the development of a "room recognizer" which is able to deduce the size and shape of a room in conjunction with a "door recognizer" which recognizes a potential door by finding two vertical edges close enough together. The long term goal of the project described here is for the robot to build simple maps of its environment, presumed to be a single floor of an office building, and to localize itself within this framework.

Thesis Supervisor: Rodney A. Brooks
Title: Associate Professor of Computer Science and Engineering

I would like to thank all the people at the lab who helped me with both technical and emotional support during the past two years.

For technical support I am grateful to David Clemens for writing and maintaining the latest incarnation of Grok, to Rod Brooks, my advisor, and Barb Moore, Michael Erdmann, and David Siegel, who sat around puzzling over equations with me, to Ian Horswill, Jon Connell, Anita Flynn, Peter Cudhea, and Paul Viola, who discussed ideas with me, to Eric Grimson and David Braunegg who thought about stereo vision with me, to Jonathan Amsterdam who helped me get started, and to Liora, Nomi, and all my friends, who went out to dinner with me and encouraged me and got me through.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the research is provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685 and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Relationship between Mobile Robots and AI

What is AI? To some, AI is the study of how the brain works, and their research is guided by this goal — that is, if a simulation seems to work differently than the brain, as evidenced by psychological observations, then throw it away. To others, AI is an engineering problem. They are concerned less with what biological intelligence is and are more concerned with the question of how to simulate it. In order to simulate biological intelligence, we have to replace the biological goals with electronic goals and have the behavior of our "being" motivated by these goals, as is a biological creature in reproducing or consuming.

Many traditional AI programs suffer from the fact that their interaction with the world is limited to that which their creator chooses to input, that which their creator deems important, and limited moreover in a highly structured way, such that the information is already in digested or preprocessed form. A program like this can therefore be criticized on the grounds that what it "knows" was not learned, but rather was a product of its creator, i.e., its knowledge is in no way independent of its creator and therefore is not intelligence, but rather the mere

byproducts of a clever program. Intelligence will not be classified as such until such an artificial creature exhibits behavior which is unpredictable and unanticipated by its creator; this creature will then be perceived as being intelligent. This behavior can only be a product of the information at the creature's disposal, and this information, if not programmed or a consequence of programming, can only be acquired during the course of the creature's non-deterministic and unforeseeable interaction with the real world.

Such, then, is one philosophical motivation for mobile robots. Although the field is far from the point where we can equip a little electronic creature with a few sensors, let it loose in the real world and expect it to fend for itself intelligently, making decisions along the way, one of the goals of the work of the mobile robot group at MIT is to build robots that will eventually be able to function like this.

1.2 The Goal of Robot Navigation

What, in a word, is robot navigation? When we say that we want a robot to navigate, we generally mean that we want it to be able to master the art of getting from place to place in a successful fashion, without bumping into things. This implies an understanding of what a "place" is, be they distinct rooms, or maybe merely points on a grid. It also requires an understanding of how places are connected to eachother, so that the robot may plan routes. On a more advanced level, navigation may require an understanding of the meaning of places in relation to specific tasks. For instance, when delivering mail from the mailroom to individual offices, what is the difference between the the starting point and the destinations in terms of the tasks to be performed at each place?

On first glance, the problem of navigation as described above seems quite straightforward to solve. After all, we are dealing with machines whose measurement capabilities are far more precise than those of humans. Why not simply give

the robot an accurate map of the local environment, a precise odometer, write one program that performs trajectory integration and another program that finds free paths between points in the map, and let the robot loose? What makes this seemingly simply problem difficult? There are several answers to this question. First of all, any drift in odometry, even the most minute, causes an error in the calculated location with respect to actual location which is unbounded with time. Secondly, the problem is further complicated by the fact that most interesting environments are not static, but are rather changing continually. For a robot to successfully navigate, it must somehow cope with the unpredictability of an object disappearing from a place that it was before, or appearing suddenly in front of its planned path.

The pragmatist will realize that the first approach to the problem was naive, and that of course the robot needs to have sensing capabilities. By using its sensors, the robot will have a sensing feedback loop with the environment, and this will solve both problems: it will be able to sense actual movement as compared to assumed movement and will therefore be able to correct its position, and it will also be able to sense any unanticipated objects in its path and so avoid them.

Much work has been done in this vein, using both vision and sonar to build maps, or, given a map, to successfully plan and execute trajectories to a goal. The most successful examples of robot navigation occurred in carefully engineered environments where the robot was able to accurately predict what its sensory input should be at any point, and correct for drift by comparing actual input to the projected input [sha84]. In unstructured environments however, the problem became much harder, and the obvious approaches failed to produce good results. Those techniques which have relied on dead reckoning and trajectory integration have been plagued with the problems of cumulative error [Tho79], and those techniques which rely on sensory input to model the environment have struggled with the inability of present sensing techniques to return information reliable

enough to conclude anything about the environment with certainty. An example of the extent of noise in typical sensor data is given in [Dru87].

1.3 A New Approach to Robot Navigation

In this thesis I have attempted to attack the problem from a different angle altogether, using the way people navigate through buildings as insight and inspiration. People, when sitting in a particular room in an office building, may be able to tell you approximately in what direction another given room is, maybe not. They can also probably tell you some sequence of rooms they would go through to get there, but they would not be able to tell you the distance in feet, nor the angles they would turn after entering any given room on the path. They definitely won't have a trajectory plan of where to walk to in order to get as fast as possible to the comfy chair based on their model of the room.

The manner in which locations are represented and understood by a navigator greatly affects its behavior. Two terms are used to characterise two different types of maps: a *metric* map is one which contains precise geometric information, such as absolute locations on an xy grid, and a *topological* map is one in which distinct places are represented as nodes in a graph, where two nodes are connected in the graph when there is such a connection between their corresponding places in the world. This latter map contains virtually no metric information. Though it is not known whether or not humans actually carry a topologic map of their local environment, their behavior suggests that they understand their environment in topological fashion [Lyn60], whereas all work in robot navigation has used the metric representation.

The goal of this project is to have a robot build simple maps of its environment (presumed to be a single floor of an office building) containing the minimal possible information, where the rooms are represented as nodes connected by paths. The

robot should also be able to determine its approximate location in this map at all times. This approach tries to circumvent the problems that arise from dynamic environments by taking note only of those features of the environment which never change, that is, the position of the walls. To put this in terminology of Chatila and Laumond [Cha85b] [CL85], the approach attempts to get straight to the "topological level" of information without first mapping the "geometric level", thereby avoiding its associated pitfalls. The geometrical information is no doubt useful for (a) obstacle avoidance, and possibly (b) room recognition, but it is not necessary to keep an accurate 3D map to perform these functions accurately. Obstacle avoidance can be taken care of by a reactive module suited for this purpose.

In addition, this entire algorithm makes use of only visual information from relatively uncalibrated instruments, which in this context means that the configuration of the cameras with respect to the robot and its motion is restricted to a small range but is not adjusted or known *a priori*.

1.3.1 Using the Topological Representation

To use rooms as the basic building blocks in a topological map and to move between them, a robot must be able to recognize both rooms and doors. The bulk of the technical work done for this thesis consisted of building two modules to do exactly that. The "roomfinder" works using rotational vision to find the perpendicular direction to the walls of the room, and then using stereo vision to estimate the distance to each wall. For this last stage, the robot looks only at the junction between the wall and the ceiling, whose location is presumed to be fairly invariant and which is rarely obstructed. From the distance to each of the four walls, the dimensions of the room can be calculated. The "doorfinder" uses forward motion and single line stereo vision to find two vertical lines, suitably separated from eachother, through which it tries to lead the robot.

1.4 Context of the Project

The work described above was motivated by the ideas inherent in the design of the subsumption architecture, a control methodology for robots which combines individual behaviors, modelled as finite state machines, into an interacting whole. The characterizing feature of this organization is the absence of a central controlling unit to guide the overall behavior of the robot; this allows for modularity in design and debugging, as well as the potential for parallel implementation. An example of decomposition of an overall behavior into modules can be found in [BC86]. To date the group has built two robots, Allen and Herbert, using this design as the core. In general, sonar has been used for obstacle avoidance, though visual algorithms for obstacle avoidance have been studied as well. The group's new robot, Seymour, is presently being designed and built, and is intended to use only passive sensing. The work presented here is intended to be integrated into this most recent robot as a higher level behavior.

1.5 Overview of the Thesis

Chapter 2 presents the theory behind the room finding sensor I have developed, and derives in detail all the equations used in subsequent chapters. Chapter 3 presents the implementation and results of the theory, and discusses problems encountered along the way. Chapter 4 presents the theory and implementation of the doorfinder, the module responsible for bringing the robot from room to room. Chapter 5 talks about problems and issues involved in building and using maps, and presents some psychological studies done on humans and how they solve these problems. Chapter 6 discusses how with the results of the two sensors, the roomfinder and doorfinder, one can build accurate and useable maps. Finally, chapter 7 discusses results of this work, related work, and how they fit together.

Chapter 2

Visual Analysis of the Room Finder

In this chapter I will present the rationale behind my approach to the problem, and theory behind the work that I have done. All of the equations that I used in the roomfinder implementation will be derived and explained.

2.1 The Approach: Breaking the Back of the Problem

When a human walks into a room, what does he see which leads him to decide that the space into which he has walked is a room, and not a corridor or a closet? What enables him to conclude that despite the presence of furniture and clutter, the space is sharply defined usually in the shape of a rectangle? Most humans have nearly infallible real-time stereo matching and can thus create a $2\frac{1}{2}D$ reconstruction of the space which they see. They can then determine where the walls are, and thus get a feeling for the size and shape of the enclosed space which they are in, ignoring those obstacles which may be standing against or blocking the walls, such as tables or cabinets.

The observation which motivates my approach to this problem is that the clearest clue to room size is the location of the junction between the wall and ceiling in all parts of the room. Even humans are hard pressed to determine the location of a wall when the location of this junction cannot be seen or inferred. Imagine a case in which a tall bookshelf is blocking the view to this junction; the observer has no information as to how far the wall is. If the bookshelf is standing between the viewer and the wall such that the ceiling wall junction can be seen, the observer will have no trouble in determining where the wall is.

A robot, with far less processing power than humans have, might be able to determine the dimensions of a room and approximate location in a room using this visual strategy. This will require several steps:

(a) Determining the perpendicular direction to the walls. This is done by taking a one dimensional picture at constant time intervals while spinning, and creating from them a two dimensional composite image. The point in time during which most of the curves in the resultant composite image reaches their maximum values is the point at which the camera was facing the wall.

(b) Pinpointing the ceiling wall junction in the direction of the walls. This is done by using stereo matching in the vertical direction to determine depth of features in both images.

(c) Using the distance to all four walls to localize itself and decide the room's size. The distance to a wall is a function of the junctions' height in the image plane, scaled by an unknown factor, which is the height of the ceiling of the room.

In the proceeding analysis the following assumptions are made:

- All rooms are rectangular.

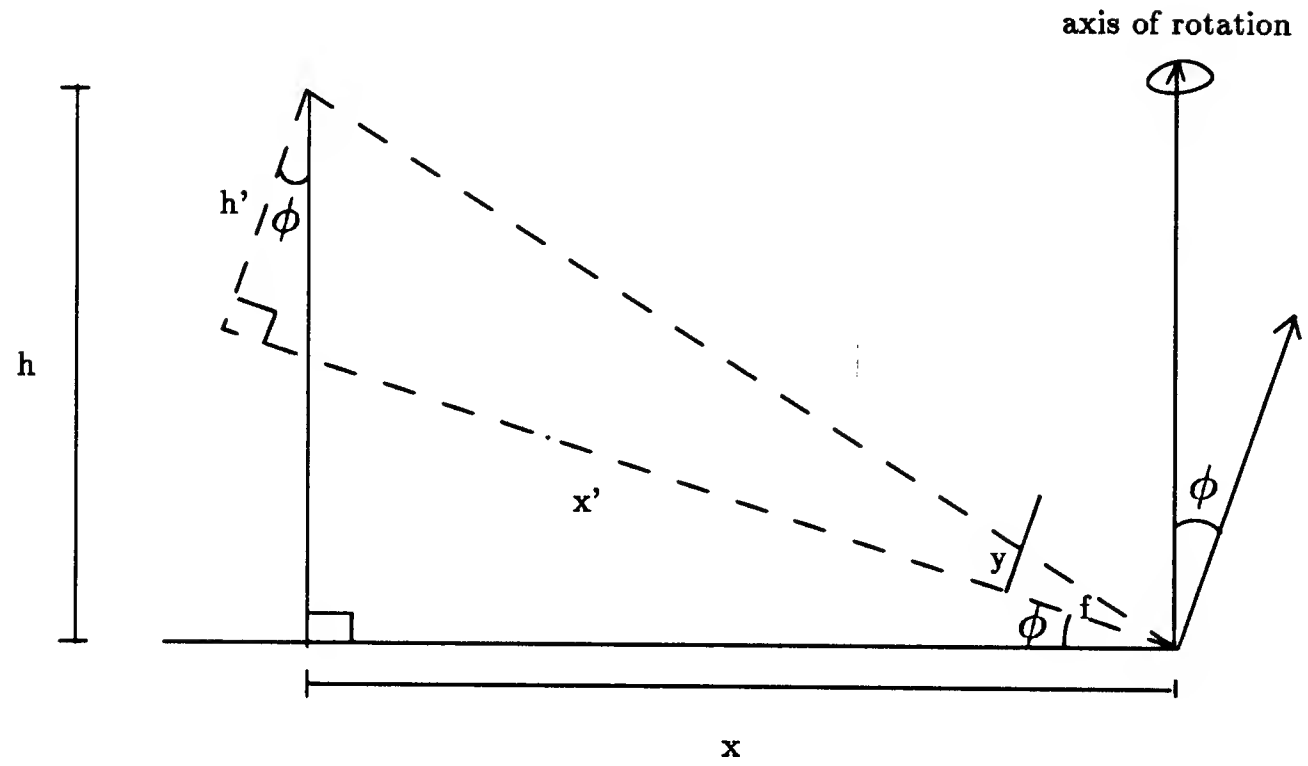


Figure 2.1: A side view of the position of the robot in the room.

- At a ceiling wall junction there is a color or illumination change which will give rise to an edge in an image.
- Ceiling height is constant throughout a room.

2.2 Finding the Perpendicular Direction and Distance to the Walls

The configuration of the camera with respect to a horizontal feature is shown in figure 2.1. The camera is tilted with respect to the wall by some angle ϕ. The height of any feature in the image plane, y, is a function of the perpendicular distance x' to the feature, the focal length f of the lens, and the height h' of the feature:

$$y = \frac{fh'}{x'} \tag{2.1}$$

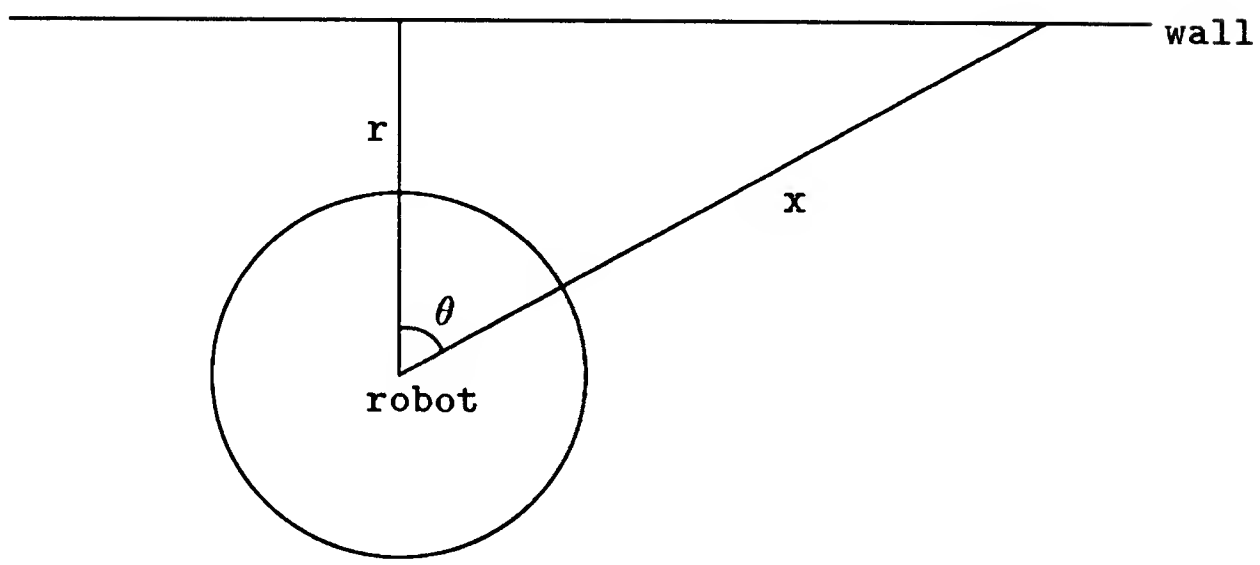


Figure 2.2: Same scene, top view.

x' is a function of the angle ϕ:

$$x' = \frac{x}{\cos\phi} + h' \tan\phi \tag{2.2}$$

And so is h':

$$h = x\tan\phi + \frac{h'}{\cos\phi}$$
$$h' = h\cos\phi - x\sin\phi \tag{2.3}$$

Note that x, h, ϕ, f are constant. Also, x is a function of the angle θ (fig 2.2):

$$x = \frac{r}{\cos\theta} \tag{2.4}$$

where r is the perpendicular distance to the observed wall and also is constant. If we rotate the camera with respect to the feature, we see that the height of the feature in the image plane is a function of θ. Taking equation 2.1 and successively substituting x', h' and x in that order, we get:

$$y = \frac{fh'}{\frac{x}{\cos\phi} + h'\tan\phi} = \frac{fh'\cos\phi}{x + h'\sin\phi}$$
$$= \frac{f\cos\phi(h\cos\phi - x\sin\phi)}{x + \sin\phi(h\cos\phi - x\sin\phi)}$$

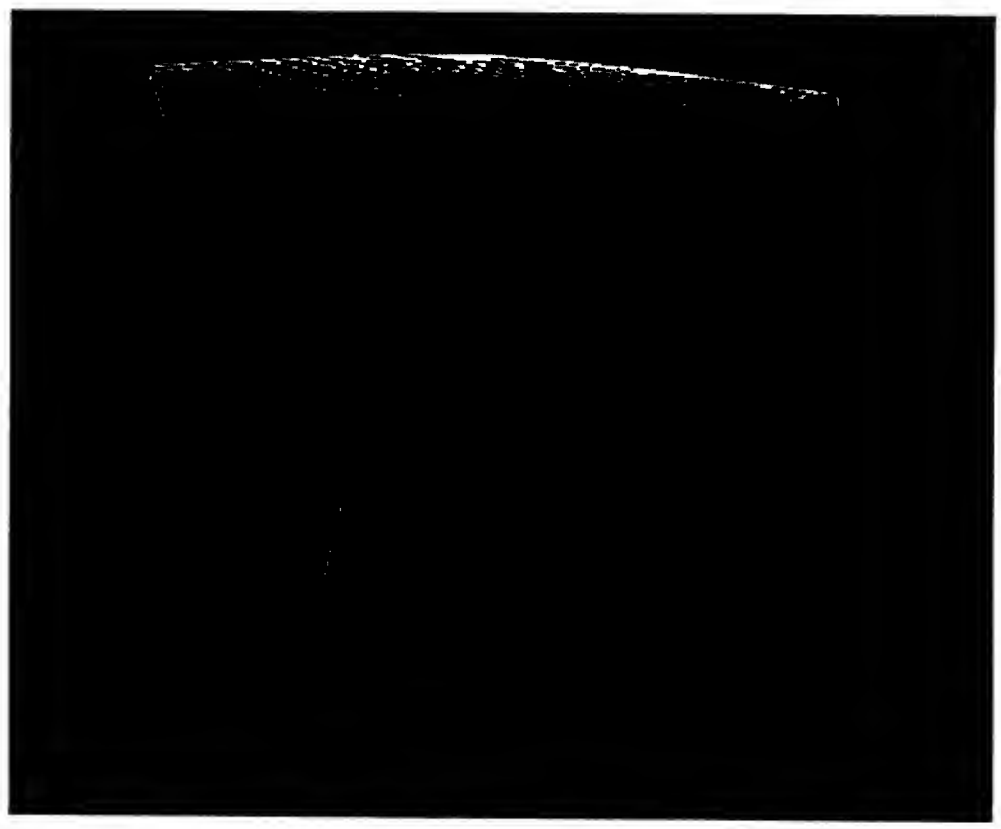


Figure 2.3: A plot of equation 2.5: y as a function of ϕ and θ. ϕ is along the axis receding to the left away from the viewer and ranges from 90° to 0°; θ is the axis receding away to the right and ranges from −90° to 90°.

$$= \frac{f\cos\phi(h\cos\phi - \frac{r}{\cos\theta}\sin\phi)}{\frac{r}{\cos\theta} + \sin\phi(h\cos\phi - \frac{r}{\cos\theta}\sin\phi)}$$

$$y = \frac{f\cos\phi(h\cos\phi\cos\theta - r\sin\phi)}{r + \sin\phi(h\cos\phi\cos\theta - r\sin\phi)} \tag{2.5}$$

In figure 2.3, y is shown as a function of θ and ϕ for a fixed r and h. Note that y, the height of the edge cause by a horizontal feature, comes to a maximum when the camera is facing the feature head on.

Next, solve for r, the distance to the feature.

$$r = \frac{h\cos\theta(f\cos\phi - y\sin\phi)}{f\sin\phi + y\cos\phi} \tag{2.6}$$

2.3 Self Calibration of ϕ

Let us for the time being make following assumptions and demonstrate their truth in the next chapter: the robot can tell when a particular camera is facing the wall by rotating, taking single line snapshots throughout the rotation and forming a two dimensional composite image from them, and then finding the points in time at which the curves in the composite image come to a maximum; it can also identify the ceiling edge in the resulting image (the latter assumption I will justify shortly). Now, in addition to f and y which are known, we also have $\theta = 0°$. There remains only one unknown on the right side of equation 2.6, the value of the variable ϕ.

Using the assumption that all of the ceiling edges are at the same height and subsequently dividing through by the constant h, the above equation becomes:

$$\frac{r}{h} = \frac{f\cos\phi - y\sin\phi}{f\sin\phi + y\cos\phi} \tag{2.7}$$

Let us name the distances from the robot to each wall r_1 to r_4. Adding opposing walls, we get:

$$\frac{r_1 + r_3}{h} = \frac{\sin 2\phi(f^2 - y_1y_3) + f\cos 2\phi(y_1 + y_3)}{f^2\sin^2\phi + f\sin\phi\cos\phi(y_1 + y_3) + y_1y_3\cos^2\phi} \tag{2.8}$$

This sum must remain constant for the same two walls no matter where in the room the robot may be; using this, we may determine the value of ϕ given two pairs of y_1 and y_3 from two different locations in the room. ϕ is the angle which yields the same value of $\frac{r_1+r_3}{h}$ for the two pairs. Figure 2.4 shows plots of equation 2.8, for ceiling edge pairs taken from actual images from five different locations in the same room. The top graph shows the shape of the function for one pair, the middle shows the function for all five pairs, and the bottom figure shows a closeup of the intersections of the functions in the relevant range, with ϕ ranging between 25°-45°. The points of intersection gives the value of ϕ. Now, f, ϕ, y_1 and y_3 are known, so the room's dimensions, scaled by the unknown but constant h,

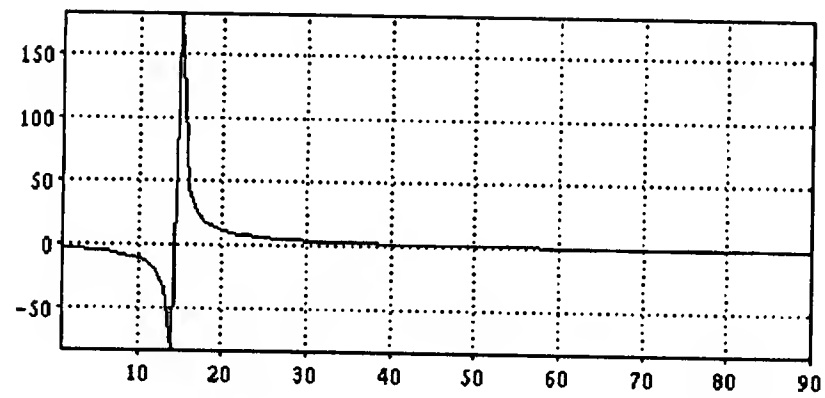
are computable and are given by $\frac{r_1+r_3}{h}$ and $\frac{r_2+r_4}{h}$. This information can be used to identify rooms by their dimensions. Also, once ϕ is known, the robot's position and orientation within the room can be determined at all times and can be used to place the location of doors with respect to the room.

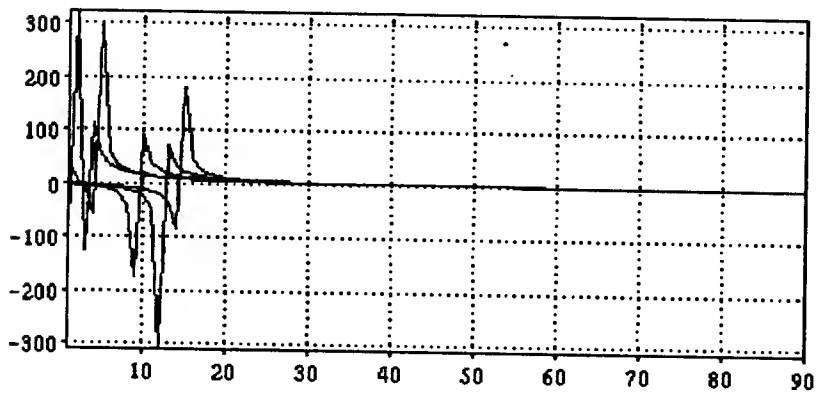
2.4 Determining Relative Depth Using Uncalibrated Stereo

The preceding discussion assumes that the robot can identify the ceiling edge, where the walls meet the ceiling. Now I discuss the method for doing this using uncalibrated stereo.

Consider figure 2.5. Two cameras in a fixed position with respect to each other determine a line which passes through the two focal points. If the optical axes of the two cameras are not parallel, then this line intersects each image plane at a point, called the epipole, for that plane. Any feature point p in conjunction with this line determines a plane, called an epiplane, which intersects each image plane at corresponding epilines. Therefore, every point in this plane projects somewhere onto the epiline for that plane in the image plane. Note that all epipolar planes go through the epipole in both image planes. If the relative position of the two cameras is known, stereo matching is reduced to a search along one dimension, the corresponding epilines, instead of two dimensions.

In order to successfully perform stereo matching in our application, we must have the two cameras placed vertically with respect to each other for the following reason: because the features that we are searching for are horizontal, they give rise to edges only in the vertical direction. A configuration in which the two cameras are displaced horizontally will have only horizontal corresponding epilines, and no features to match along them. It is necessary that the epilines be orthogonal to the features being matched; therefore, the cameras must be placed vertically with





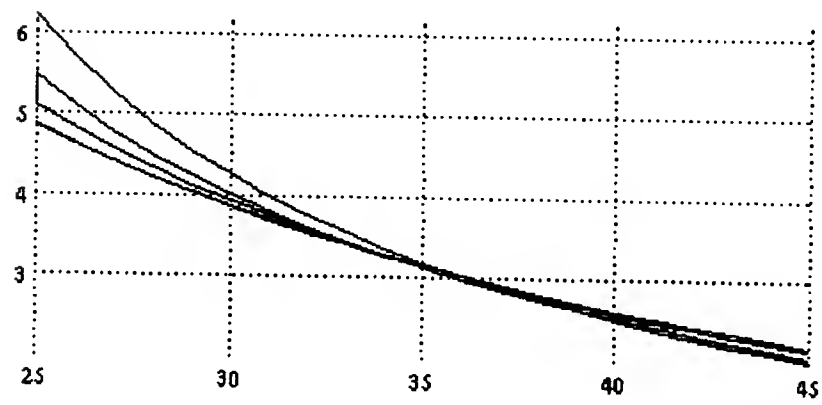


Figure 2.4: Equation 2.8 for five ceiling edge pairs. $\frac{r_1+r_2}{h}$ on the y axis is plotted against ϕ on the x axis. $f = 4.8$ for both curves. The first graph is the equation for one pair, the second is a plot of the function for all five pairs, and the last graph shows the function for the five pairs in the range $\phi = 25°$-$45°$. The five pairs yield 10 intersection points, whose average intersection value occurs at $\phi \approx 35.5°$.

respect to each other.

Since the camera configuration is not carefully calibrated, there may be some misalignment between the two forward pointing cameras; therefore there is no guarantee that the two optical axes intersect at a point. However, as the robot spins, each of the two axes sweeps out a cone, and the intersection of these two cones forms a circle. Since we have identified the point in each image where that camera was facing a wall, we have identified a "virtual" epipolar plane – one which occurs in both images, but not coincident in time. The process of finding the corresponding walls in the two images and matching along those corrects for any misalignment. Therefore, the epipolar analysis applies in this case.

In order to correctly determine the value of h for a given feature, we must have accurate knowledge of the baseline separation and the relative angle between the two cameras. But, for the previous computations, we did not require that h be known, only that it be possible to pick out the ceiling edge so that we could use its corresponding image height. The key to finding the correct edge lies in the fact that the ceiling edge will always be the edge farthest from the camera and is distinguishable by this. Weinshall [Wei87] discusses an algorithm for determining relative depths of observed features using uncalibrated stereo; this method works for a limited range and unfortunately breaks down for our case, in which the observed feature is actually contained in the epipolar plane. However, the ideas presented there served as inspiration for the following work.

The term "horopter" refers to the locus of points in space which, when fixated upon from two points separated by some baseline distance, have zero retinal disparity. Assuming perfect geometrical conditions (which biology does not abide by), the horopter lies on a circle [BKT86]. Taking this same idea, but working with two cameras which have a fixed fixation point, we can define lines of equal disparity. Figure 2.6 shows the configuration of the two cameras, and some point in the epipolar plane with coordinates (x, y). These coordinates are given with

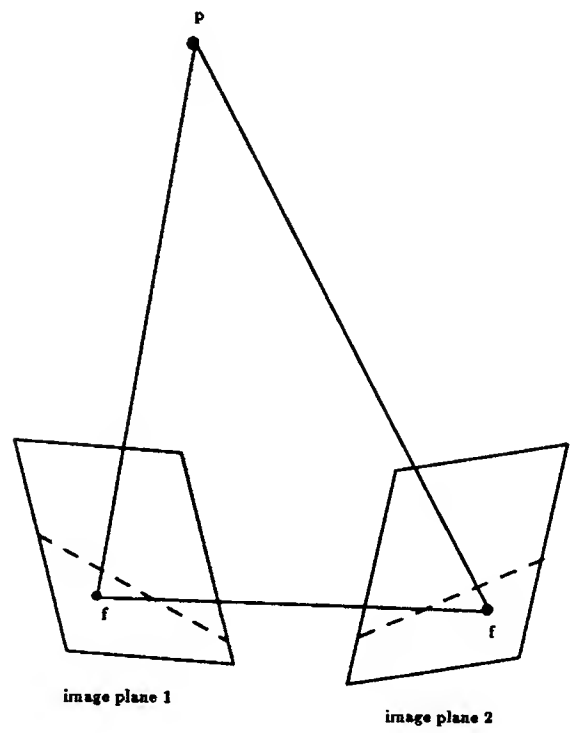


Figure 2.5: All points in an epipolar plane will project onto the same pair of epipolar lines in the two images.

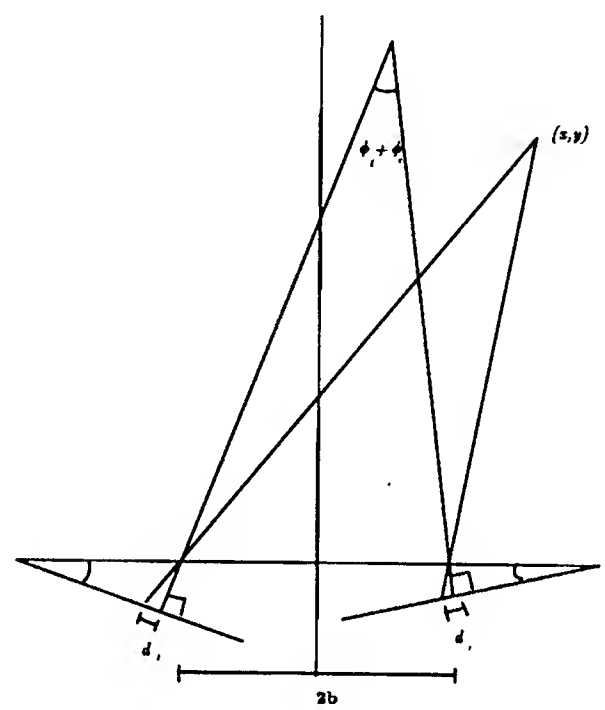


Figure 2.6: Configuration of two cameras whose optical axes intersect at a point.

respect to a reference frame whose origin is the midpoint of the segment connecting the two focal points, and whose x axis lies along this same segment. The expression for the disparity of the point is:

$$d_l - d_r = \frac{f_l x_l}{y_l} - \frac{f_r x_r}{y_r}$$

$$= f_l\left(\frac{\cos\phi_l x - \sin\phi_l y + \cos\phi_l b}{\sin\phi_l x + \cos\phi_l y + \sin\phi_l b}\right) - f_r\left(\frac{\cos\phi_r x + \sin\phi_r y - \cos\phi_r b}{-\sin\phi_r x + \cos\phi_r y + \sin\phi_r b}\right)$$

Making the simplifying assumption that $\phi_l = \phi_r$ and $f_l = f_r$, we get the much more manageable expression for the curves of equal disparity:

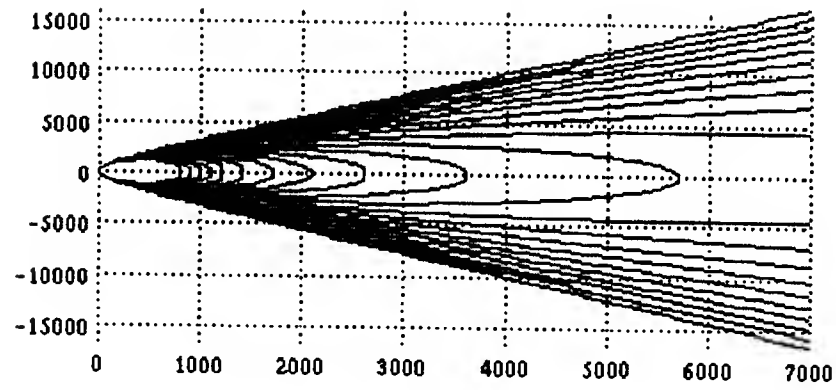
$$\frac{k}{f} = \frac{(b^2 - x^2 - y^2)\sin 2\phi + 2\cos 2\phi by}{y^2 + \sin 2\phi by + (b^2 - x^2 - y^2)\sin^2\phi} \tag{2.9}$$

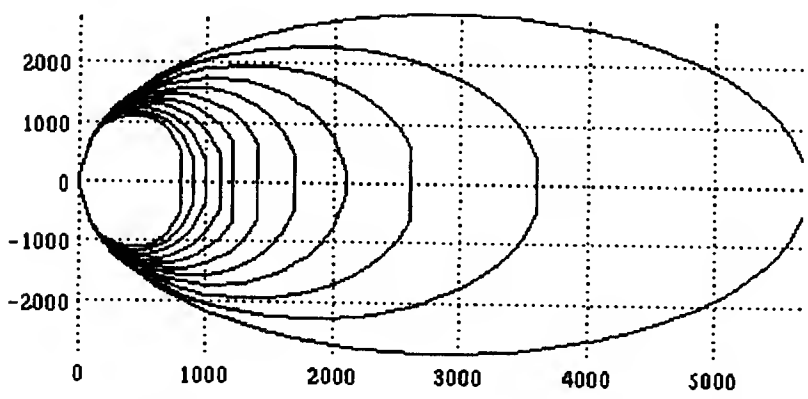
where k is disparity, and is constant in this equation. Figure 2.7 shows a plot of this function for $\phi = 1°$, $f = 4.8$ cm, and k ranging from -0.1 to 0.1 cm in the image plane. The curves emanate outward from the two focal points in order of decreasing k. Note that in the case where $\phi = 0°$, the above equation reduces to:

$$y = \frac{2bf}{k} \tag{2.10}$$

which describes the better known parallel camera configuration, where the curves flatten out to lines, and disparity is monotonically decreasing with depth (y).

Figure 2.8 shows the curve which passes through the ceiling edge, and the curve which is tangent to the wall and ceiling. Points b and c are at a maximum, and x is the point with minimum disparity between these two points. Assuming we are able to accurately match edges along the wall and ceiling, the pattern of disparities will first increase, dip downward to a local minimum, then increase temporarily and ultimately decrease again. The point at the local minimum is the ceiling edge we are looking for.





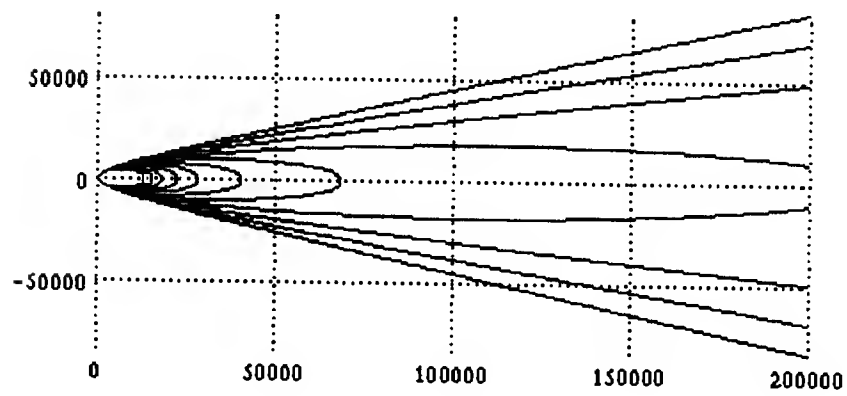


Figure 2.7: Equidisparate curves for $\phi_r = \phi_l = 1°$. The axes are switched, with the x axis running up and down. The focal points are on the the x axis. In the top figure, k ranges from -0.1 to 0.1, the middle figure is a closeup of the range 0 to 0.1, and the bottom figure shows the range -0.2 to -0.1.

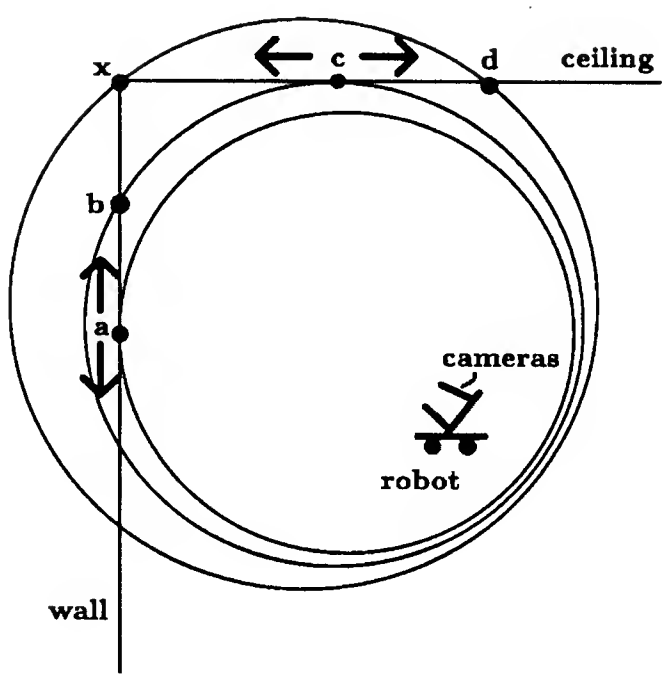


Figure 2.8: Arrows point in the direction of increasing k.

Chapter 3

The Roomfinder: Implementation and Results

The preceding chapter presented the theory behind the roomfinder. This chapter will present different methods of implementing the ideas presented there, and the results of each strategy. In particular, much of the work that I present will be variants of solutions to the $1D$ matching problem, which ultimately proved to be the hardest and most problematic stage in the correct functioning of this module.

3.1 The Robot

The initial setup is as follows: two Pulnix cameras with $4.8mm$ focal length lenses are mounted on the robot, side by side on a bar emanating from the center top and tilted backwards. The configuration is shown in figure 3.1. The cameras are assumed to be mounted roughly parallel; the procedure is insensitive to errors introduced by small deviations in roll, pitch and yaw. The bar is tilted backwards sufficiently such that the junction between the wall and the ceiling, which from now on will be called the "ceiling edge" or "ceiling-wall junction", is visible from all parts of the room. The $4.8mm$ lens has an angular range in the vertical direction

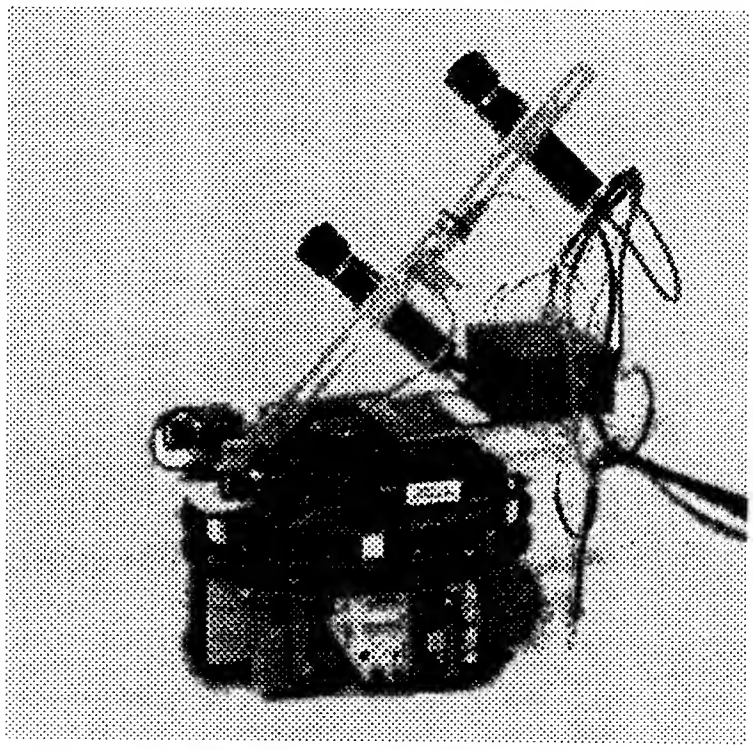


Figure 3.1: The position of the cameras mounted on the mobile robot.

of approximately 60°, and introduces some distortion into the measurements. In these experiment the distortion was roughly corrected for, since it accounted for errors of up to 20% towards the edges of the image. The image plane is $6.6mm \times 8.8mm$, which translates into 454×576 pixels. The robot can rotate, though not at a known velocity. Upon startup, the robot does not know its initial position or orientation in the room. The visual data collected by the mobile robot is processed remotely on a Lisp Machine dedicated to this purpose.

3.2 Finding the Perpendicular Direction to the Walls

The motivating assumption behind this approach is that the horizontal edges in rooms tend to be aligned with walls. As a result, in a rotational scan of the room, curves formed by horizontal edges will cluster around certain values, and these values will indicate the perpendicular direction to the walls.

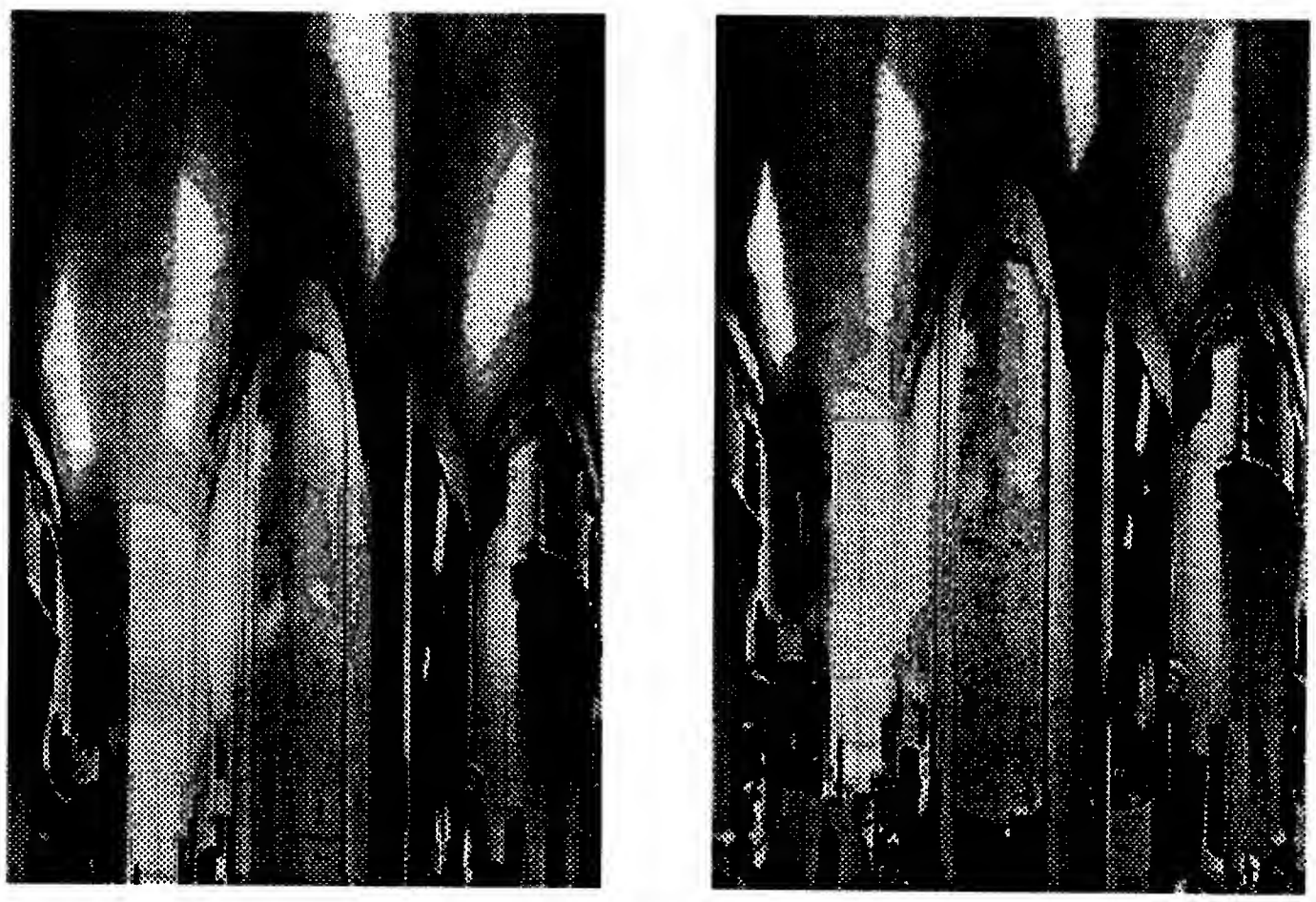


Figure 3.2: The composite image from the two cameras. This can be thought of as height, on the y axis, as plotted against time or θ on the x axis. The white deformed squares in the top part of the image are the lights on the ceiling; in the right image in the bottom right-hand corner, a Lisp Machine terminal can be distinguished. Above the terminal to the right is a bookshelf. See [BBM87] for previous work in composite images.

An experiment was performed in four very cluttered rooms in the AI lab. The robot rotated through an angle of 360°, recording the intensity values from a one dimensional strip from the center of each camera through time, and finally composing a single image from each camera. The resulting composite is shown in figure 3.2. The horizontal edges in the scene give rise to the curves in the composite image, whose shape is as predicted. In the images pictured here, the robot was not rotating at a constant velocity, but succeeding versions of the experiment did rely on constant rotational velocity.

The composite image is then convolved with the derivative of a gaussian in the

vertical direction. This performs both smoothing and differentiation in one step, and the maxima and minima of the result yield the vertical edges. The edges are then tracked through each time slice to find the curves (fig 3.3).

The edge-tracker was designed to work in real time as the image was being collected, and so works in one time slice at a time, referencing at most the previous and present time slice at any point in time. It is extremely simple and works as follows: for each edge in the previous time slice (y direction), the current time slice is scanned for a matching edge within a fixed sized window. The window is searched from the center outward, and the first possible matching edge is chosen and linked to the previous edge. An edge matches one in the adjacent time slice if its intensity gradient is in the same direction, that is, dark→light or light→dark; due to AGC (automatic gain control built into the camera) we cannot demand that the regions which the edges separate have the same intensity as in the adjacent image. The slope of the line connecting the two edges is used as the basis for determining the location of the center of the window for the successive time slice. There is also an instance variable called "variance" for each curve, which indicates how closely the last edge added to the curve fit the projected value for that time-slice; two curves contending for the same point use this variance as the criterion for determining who is the real owner of the disputed point.

Sometimes, faint edges do not show up for several time slices. In order to compensate for the lost edges, the edge tracker receives as a parameter the number of images that it is willing to skip between losing track of a curve and picking up the other side of it. The value of 2 seemed to give the maximum number of true edges, and the least number of false ones, though this value is dependent on the particular rotational velocity that was used.

The traced curves are then smoothed, and the angle at which the maximum of the curve occurs is found. This angle is the body rotation at which the distance from the robot to the feature which gave rise to the curve is a minimum. All

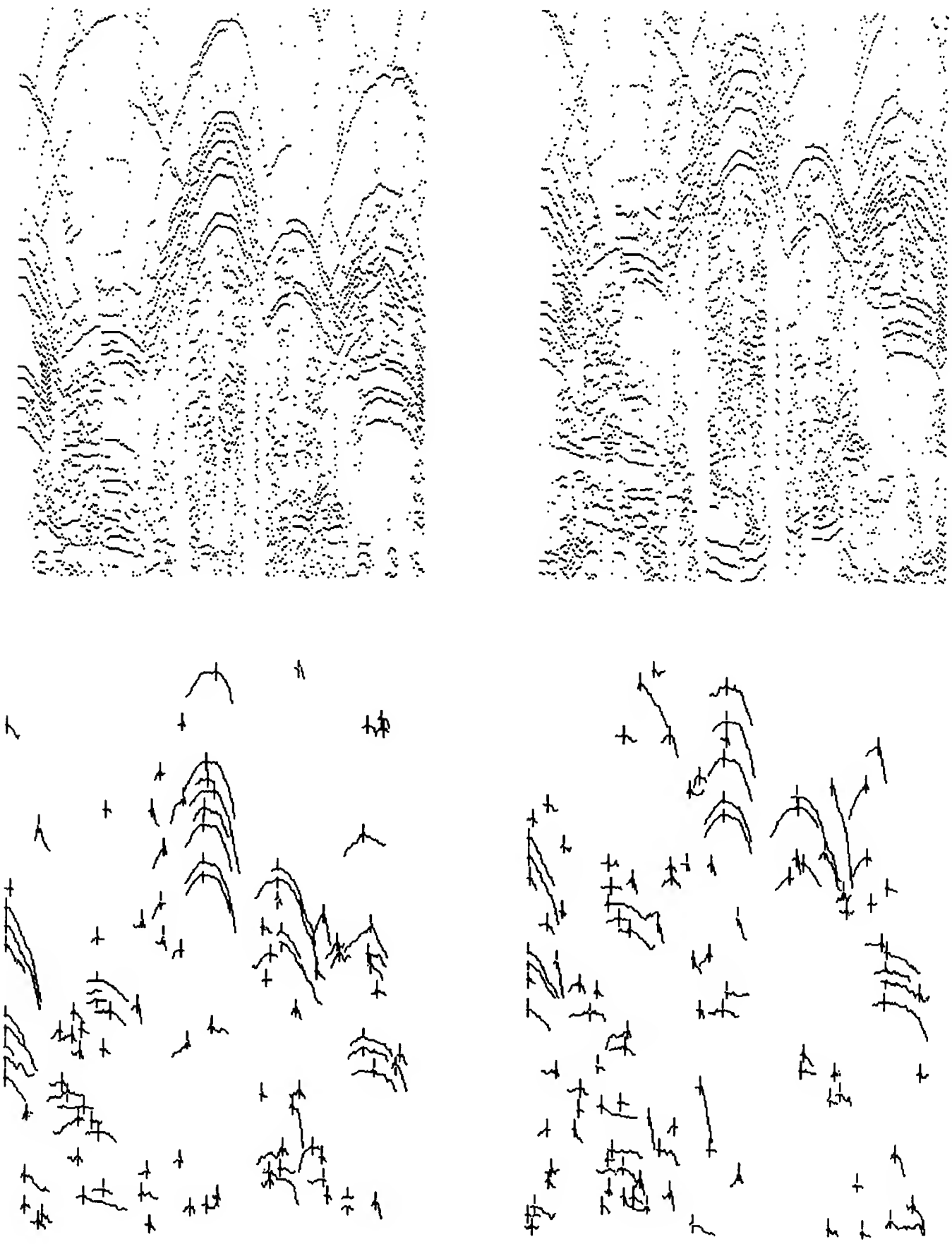


Figure 3.3: Edge-finding and curve-tracing

curves which come to a maximum are then histogrammed together in the following manner: each curve over a certain length threshhold contributes a weight of 1 to a bucket according to the x coordinate (time) of its maximum, and the bucket contents are then smoothed in order to coalesce maxima which lie within a small range of each other. Finally, the histograms from the two camera composites are cross checked against each other to find the walls. Since the velocity is not constant, the walls do not occur at fixed intervals in the images. The entire process is shown in stages in figure 3.4. The information required for this stage to work is the sweep range of the image, i.e., the robot need only be able to determine whether or not it has made a full revolution. This is usually easily available from odometry.

For the histogramming stage described, several different weighting schemes were experimented with in order to improve the estimate of the perpendicular direction to the walls and eliminate noise:

- Weighting proportional to curve length – this was in order to favor the longer edges such as tables and blackboards, which tend to line up with walls, over shorter edges like chairs and randomly placed furniture, which tended to cause noise.

- Using only the longest 1/4 of all the curves, all of which contribute a weight of 1.

- Using only curves which also have a pronounced rising or descending stage, all of which contribute a weight of 1.

While each of these strategies worked particularly well in specific cases, they all failed in some cases. The simplest strategy, that in which all curves contribute a weight of 1, worked best most consistently and never failed drastically.

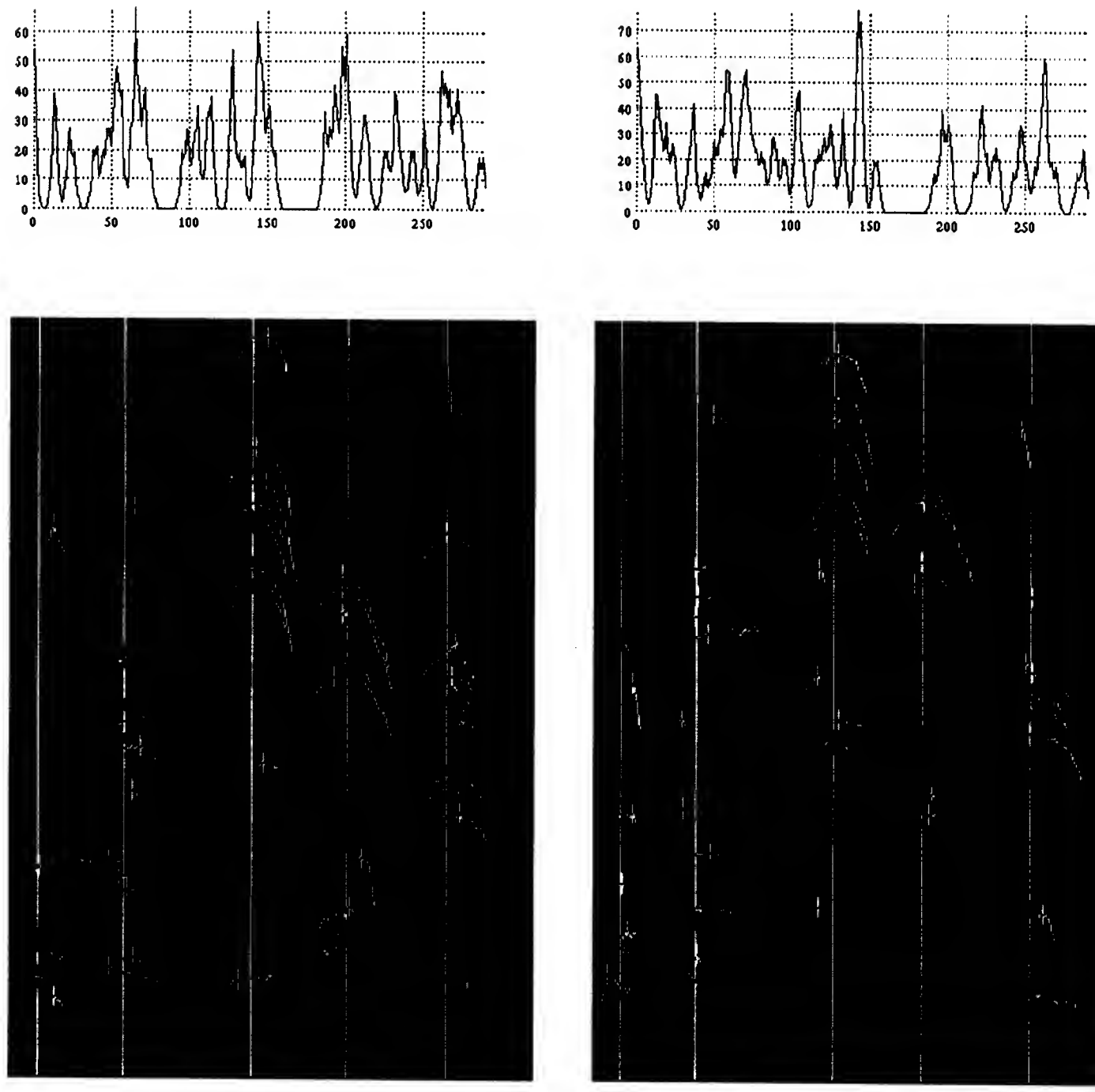


Figure 3.4: Histogramming the maxima and finding walls by cross-correlation.

3.2.1 Experimental Results

The results of this stage are consistently accurate. In tens of experiments, the algorithm has seldom failed. Additional robustness has been added by the recent ability of the robot to rotate at a constant velocity, and to rotate to specified relative angles from the current position. This has enabled the robot to identify the locations of all four walls in a room as long as the above algorithm has yielded one reliable wall.

Figure 3.5 shows the walls found when the above algorithm is run on three other rooms of varying shapes and sizes. At the time these composite images were taken, the robot was not rotating at a constant angular velocity. In addition, there were people roaming around the room, which demonstrates the insensitivity of this stage of the algorithm to the dynamicism of the environment.

3.3 Determining Distance to the Walls

As explained in the previous chapter, finding the perpendicular distance to the wall once the perpendicular direction is known requires the following steps:

- Matching edges along a vertical strip from each camera,
- Choosing the ceiling wall junction using the disparity information from the previous step, and
- Plugging the height of this edge in the image plane into equation 2.7 to get the distance to the wall scaled by the height of the room. We always use the height from the same camera for this step, though which camera we choose is unimportant.

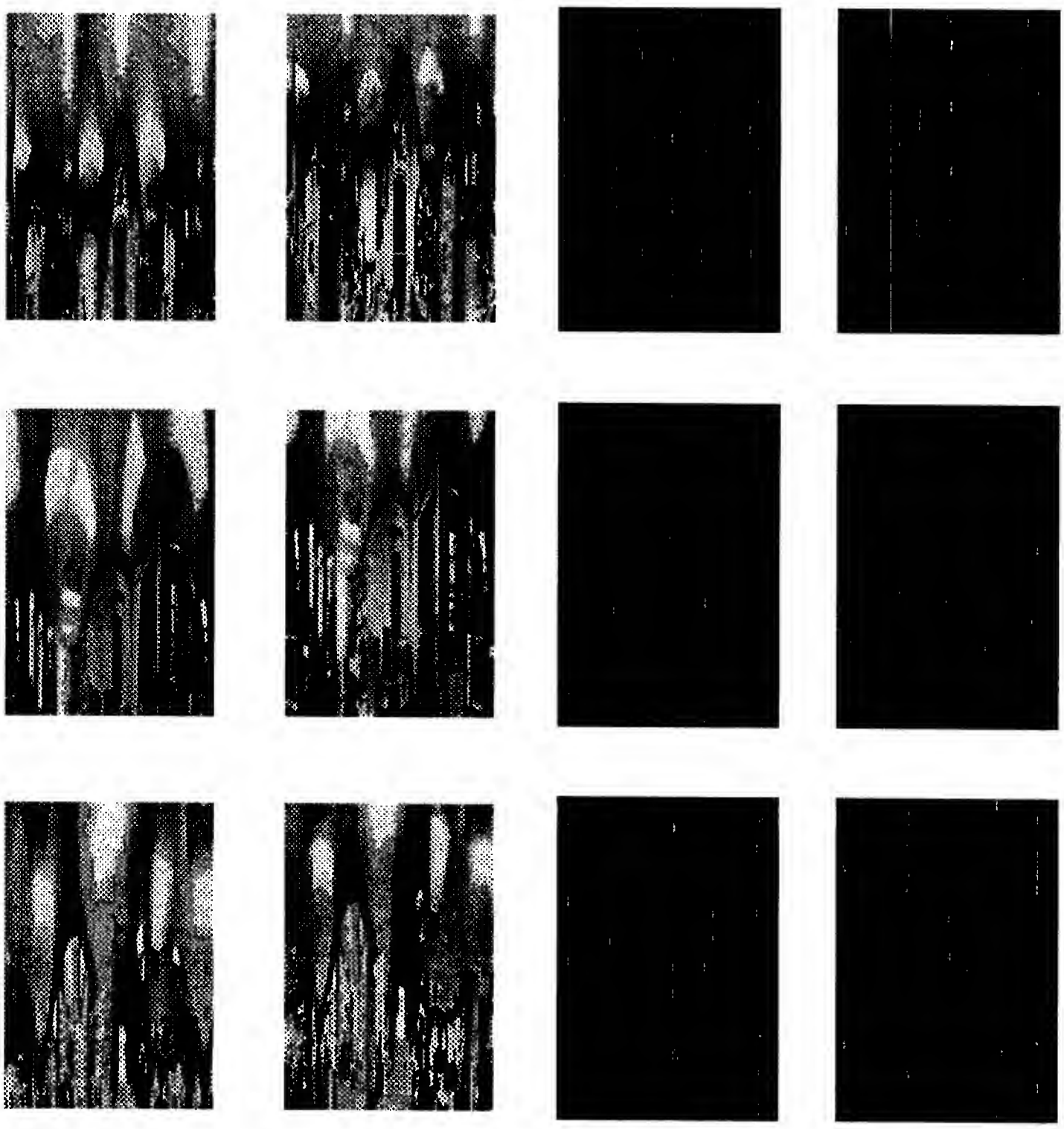


Figure 3.5: Finding walls for three other rooms.

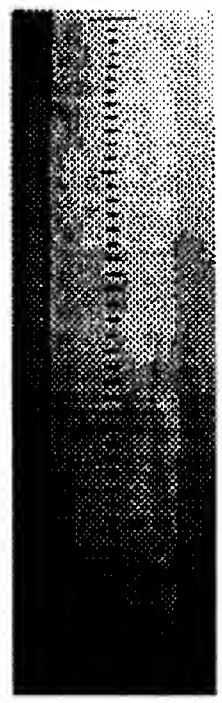


Figure 3.6: The lines pictured here are all the same distance apart in actuality, illustrating how the wide angle lens introduces distortion into the image.

3.3.1 Correcting Lens Distortion

In the actual images, it was found that the wide angle lenses distorted the image greatly, introducing errors of up to 20% towards the edges of the image plane. Figure 3.6 shows an example image from one one of the cameras; as can be seen, the image is stretched out towards the center of the image and is shrunk on the sides.

In order to accurately calculate disparity from the matching stage, the distortion must be corrected for, but it is sufficient to correct only for the $1D$ vertical strip from the center of the image. This was done in the following way: a snapshot of the same scene as pictured in figure 3.6 was taken from both cameras, and the vertical edge finder was applied to the middle vertical strip of each image. The dark→light edges from each strip were entered onto a list for each image.

Next, a function of lens distortion as a result of vertical pixel position in the image was calculated by plotting $b - a$ as a function of $\frac{a+b}{2}$ for every pair of edges with vertical pixel coordinate a and b on each list. The two functions (one per

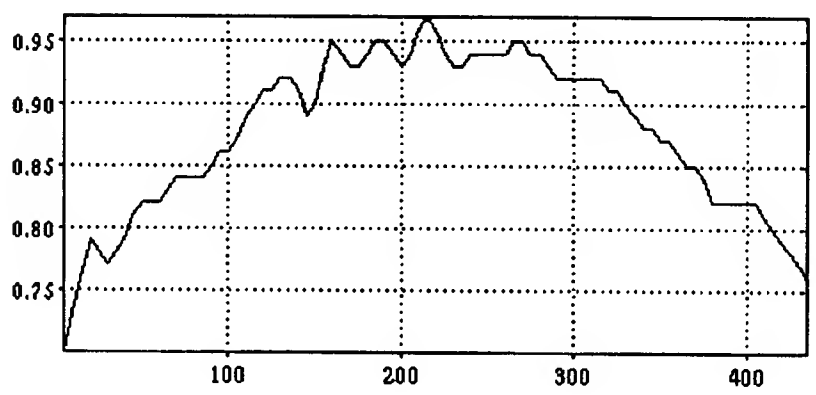


Figure 3.7: A plot of distortion as a function of vertical pixel position.

camera), represented by lists of points, were then merged together and smoothed, then normalized by dividing all values by the value in the center of the image (pixel 227) to produce the function shown in figure 3.7. This function was recorded in a lookup table in discrete intervals of 5 pixels.

The offset correction function then linearizes all distances by taking a vertical pixel offset as input and expanding every 5 pixel interval between the vertical center of the image and the offset given as input by a factor of $1/f$ for that position, where f is looked up in the table. This correction expands the image plane in the vertical direction to $7.4mm$ instead of $6.6mm$, and accounts for a corresponding virtual focal length of $\frac{3.7}{\tan^{-1}\frac{59^\circ}{2}} \approx 6.5mm$.

This correction function is used to correct the values of all y offsets before they are used in any calculations, and the adjusted value of the focal length was also used for all calculations.

3.3.2 Stereo Matching

An in depth analysis of matching algorithms is beyond the scope of this thesis, however, experimenting with variants of existing techniques proved to be a major part of the entire project, since the success of the approach rides overwhelmingly on the accuracy of this single step. I discuss some of them here.

Matching Constraints

When matching features across two images, several constraints are utilized to prune the search space of possible matches. These are:

- Epipolar – this constraint was explained in the previous chapter, section 2.4, and allows for searching for a particular feature's correspondence along a $1D$ strip instead of a $2D$ plane.
- Ordering – this constraint means that all features are present in both images in the same sequence.
- Continuity – i.e., the disparity of nearby matched points cannot be too different.
- Orientation – this prevents a point which comprises an edge with slope $+1$ from being matched with a point belonging to an edge with slope -1, for example.
- Contrast – matched points must divide regions of similar intensity or gradient strength.

In the matching problem at hand, only the epipolar and constrast constraints apply, and even these apply only loosely. The ordering constraint does not strictly hold, since the two cameras are displaced with respect to each other and therefore see different features at the two edges of the image planes. The continuity constraint does not apply, since the features at which the cameras are looking are discontinous. The orientation constraint does not apply; one need only look at figure 3.2 to see that in one image, the top of the lisp machine forms an upward sloping curve and in the other, a downward one. This is due to the baseline separation of the two cameras in the vertical direction.

Even the epipolar constraint and the constrast constraint apply only loosely, since when the calculated direction to the walls are off by only a few pixels, we are

no longer using strictly epipolar lines. The contrast constraint is violated slightly by the fact that since the two cameras register slightly different features, if a light is present in one image and absent in the other, the intensity values of the entire image are affected. However, even with these problems we can use these last two constraints effectively with some readjustments. The most obvious case of the contrast constraint was used in every technique – the constraint which prevents a match between a light→dark edge and a dark→light edge.

The scale space method of matching was considered here [Mor77], but it worked poorly, probably because of the violation of the ordering constraint, and after several attempts this line of investigation was abandoned.

At the heart of every matching technique experimented with was the Ohta and Kanade dynamic programming method, which tries to maximize the number of matches across the two $1D$ lines subject to some constraint based weighting method to assign positive values for the goodness of any individual match, and penalties for skipped (unmatched) features [OK85]. The trick in using this method was to try to override the rigidity of the ordering constraint (which does not allow for unmatched features) with high weights for very good matches according to some other criterion.

In all the matching techniques, a kind of continuity constraint was imposed by the presence of a parameter called *max-disparity*, which limited the disparity in pixels of any proposed match. The value of this parameter affected the accuracy of the different matching methods in varied and often unpredictable ways.

Matching: First Pass

The first two methods experimented with were assigning goodness of matches based on (1) comparable intensity, and (2) comparable gradient strength, with a moderately large value for *max-disparity*. Both of these methods performed quite badly. Figure 3.8 shows the matches yielded by this method for the second

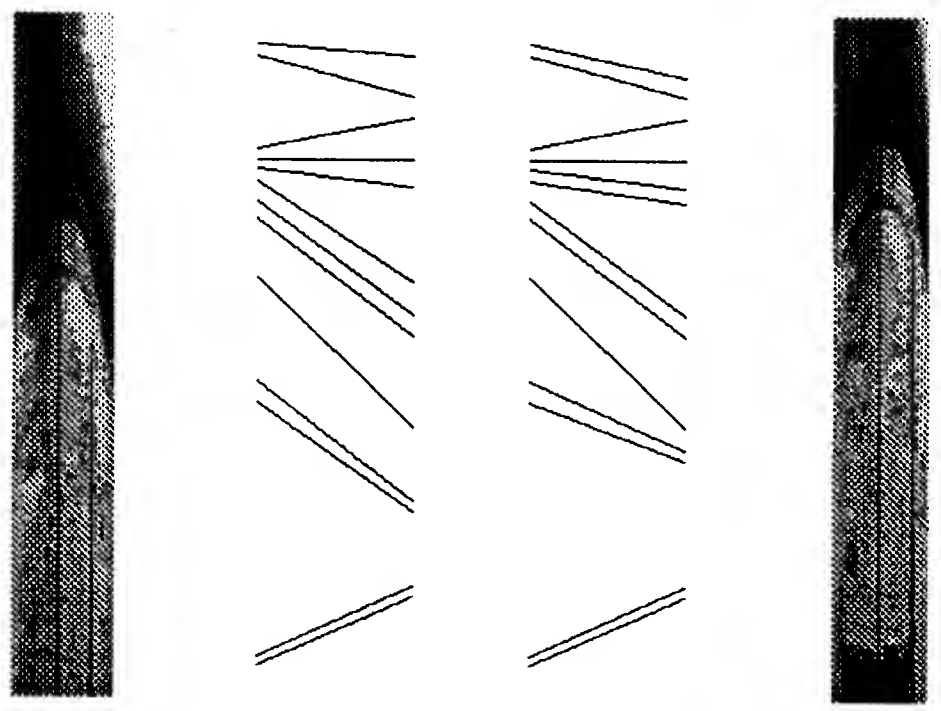


Figure 3.8: The farmost right and left illustrations are the second wall from the left and right composite images shown in figure 3.2. The vertical black line in each image is the point at which the robot has determined it was facing the wall head on. The two center illustrations show the correspondences along the two black lines: for any particular line, the left and right endpoints connect corresponding edges alonge the vertical black lines in the left and right composite image. These two illustrations show correspondences by intensity (left) and gradient strength (right).

wall of figure 3.2. As can be seen, the matches are completely unconstrained in direction; what seems to be needed is a stronger continuity constraint, along with some constraint that would incorporate the notion, apparent when looking at the two images, that the two original images seem to be vertically offset by a fixed amount.

Second Pass

The next method tried to improve upon the previous technique by finding the optimum vertical offset between the two images that would allow most of the edges to find a match within a very small *max-disparity* range. The algorithm

to do this is: every possible match between edges in each line defines an offset which is given to the matcher. If there are n edges in each line, this defines n^2 offsets, thus the matcher is invoked that many times, and each time returns a list of matches for that offset, plus some overall "goodness" rating for that set of matches based on minimizing the summation of disparities over all matches. The offset which produced the best set of all n^2 sets of matches is chosen to be the optimal offset, and then the matcher is run again with that offset in order to adjust the matches using first the intensity criterion, then the gradient strength criterion. This method yielded much better results (figure 3.9), though none of the three methods (by position only, position and intensity, and position and gradient) performed obviously better than the others, and all still had some incorrect matchings.

Using this same idea of first finding the optimal offset and then adjusting by another criterion, a fourth technique was attempted. This one attempted to find corresponding features not by absolute gradient strength, but by relative gradient strength within a small region. The results were very accurate on several test cases, and an example is shown in the fourth box of figure 3.9.

Final Matching Results

In actual runs with the robot, the carefully designed matching by relative gradient method with constrained disparities failed in an unanticipated way. Figure 3.10 shows an example where the presence of many strong edges on the bottom of a nearby wall pulled the optimum offset very high, to about 89 pixels; thus, the ceiling edge, which is further away from the robot and thus has a much smaller disparity, was incorrectly matched and thus unidentifiable. This example made clear the way in which this algorithm tends to "focus" the cameras on the part of the image with the strongest edges; if the ceiling is not in this range, it cannot be seen.

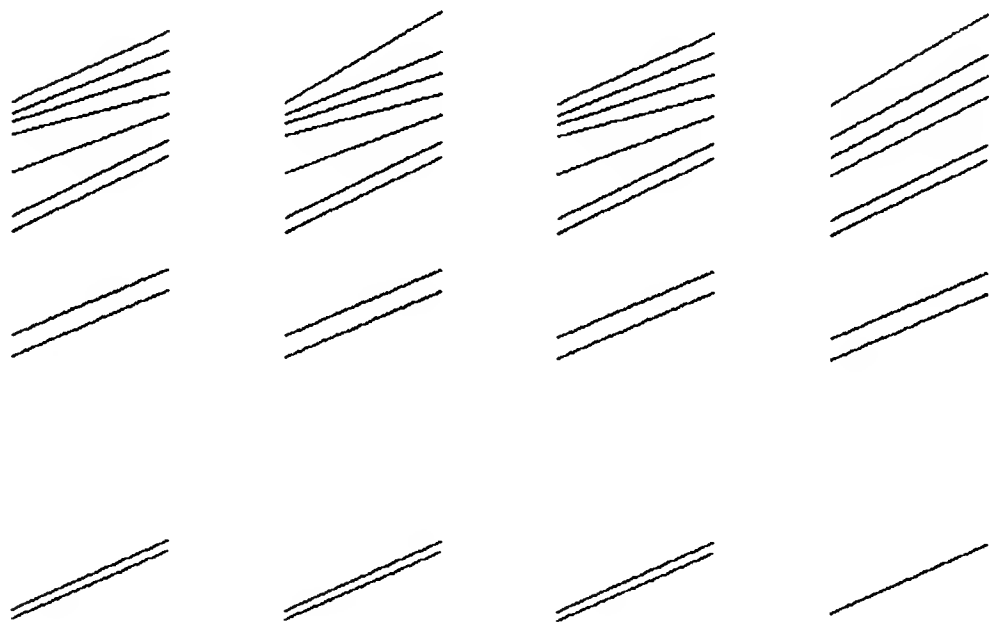


Figure 3.9: Matching along the second wall of figure 3.2 by (1) position alone, (2) position and intensity, (3) position and gradient strength, and lastly (4) position and relative gradient strength.

To fix this problem, it was decided to manually adjust this "focal range" of the robot to a range in which the ceiling edge was likely to be, which for this camera configuration turned out to correspond to a vertical offset (shift) of ≈ 30 pixels, and to expand the *max-disparity* parameter so that the robot could match edges through a much larger depth range. Since the widening of this parameter led to some more incorrect matchings, a pruning technique was added to remove outliers, with the assumption that the disparities of all matches that lie on the ceiling or wall should form a smoothly descending-ascending pattern. Both matching by intensity and matching by relative gradient yielded good results using this modification, and the results are shown in figure 3.11.

Since the relative gradient and intensity methods still did not return exactly the same matches, another pruning method was tried where only those matches that were found by both techniques were kept, and those that were found by only one of the two techniques were thrown away. This tended to throw away too many

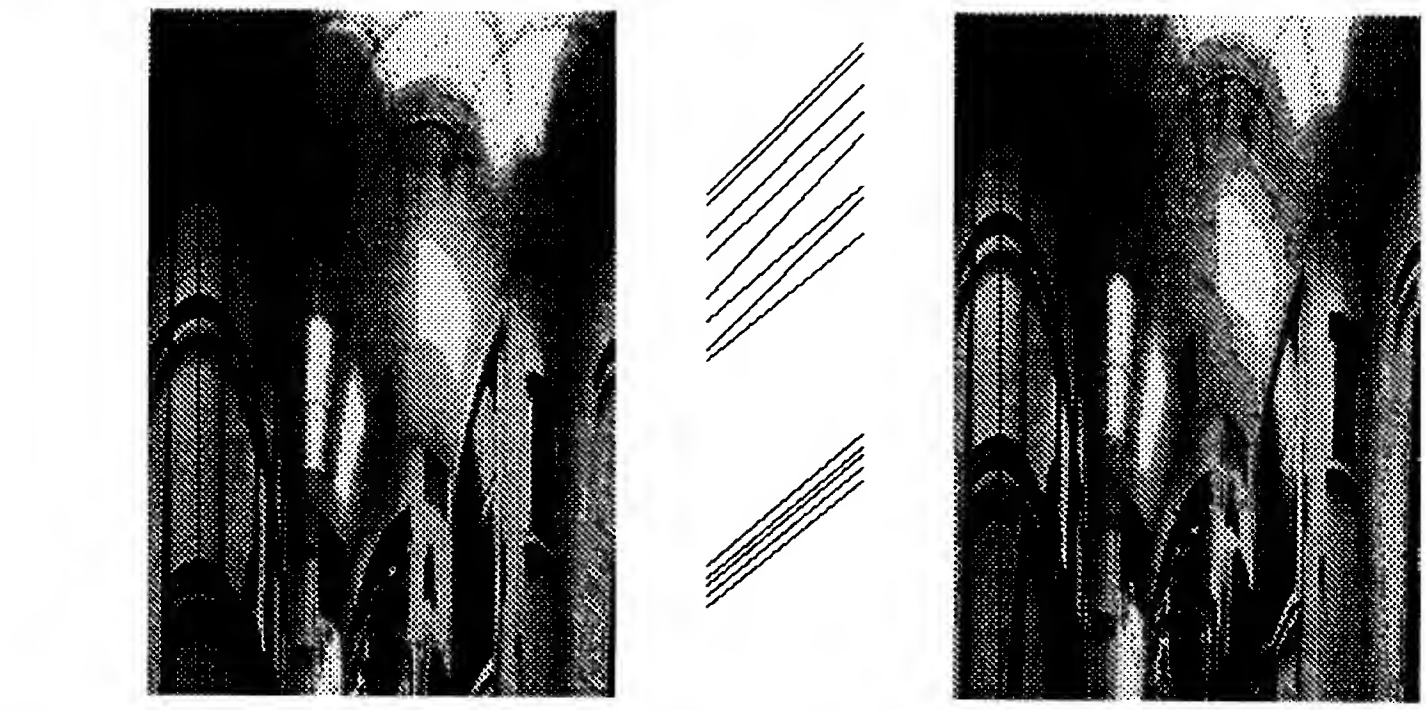


Figure 3.10: The relative gradient strength matching technique fails in this image. The black vertical lines show the walls along which the algorithm is matching. As can be seen, the presence of many strong edges on the bottom part of the wall pulls the offset very high, and so it cannot correctly match the edges on the top part of the image where the ceiling edge lies.

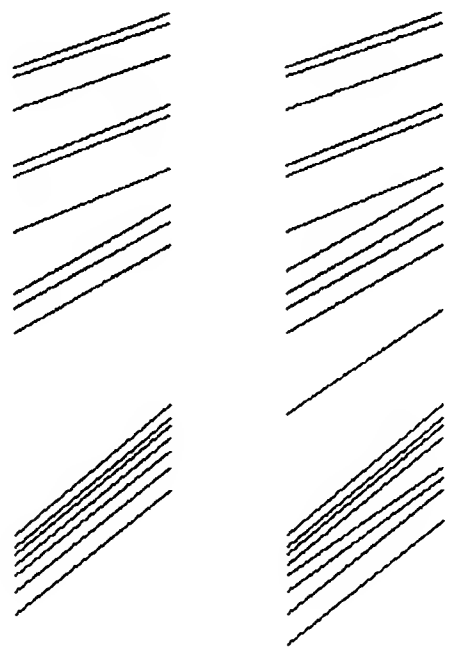


Figure 3.11: The results of the final and most reliable method, using a manually adjusted optimum offset and large disparity range. The first set shows matching by relative gradient strength, the second shows matching by intensity.

adequate matches. Finally, matching by intensity in conjunction with a manually specified optimum offset and large *max-disparity* was chosen as the best matching technique.

Even with this "best" method, chosen from all the other candidates, the matches returned on a single snapshot from the two cameras were still unreliable. Interestingly enough, when 100 snapshots in a row were taken from the two cameras in exactly the same position and lighting conditions, there were almost as many variations on different matches returned. To increase the chance that the ceiling edge chosen from any particular set of matches was the correct one, the sets of matches returned from consecutive image pairs were allowed to vote for best ceiling edge, and the winner was deemed to be the actual ceiling edge. Using this method, an accuracy rate of about 75% was achieved over about 50 trials.

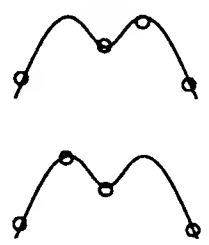


Figure 3.12: Though the disparities of matches along two perpendicular lines follows a two-humped pattern, it may be the case that there are not enough matches to elucidate the position of the ceiling edge in this pattern. These two examples are different, but are indistinguishable from each other when looking only at the disparity pattern.

3.3.3 Identifying the Ceiling Wall Junction from Disparity Patterns

As discussed in the previous chapter, the pattern of disparities should elucidate the location of the ceiling wall junction, since this junction should fall into a local minimum well. In actual experiments however, it was found that there were not enough matches to constrain the position of the ceiling edge to a unique position in the resulting disparity plot. For instance, in figure 3.12, the ceiling edge is the first edge before the maximum disparity value in the top example, and the first edge after the maximum disparity in the bottom example. The case in which the desired junction is the maximum disparity pair occurs when there are no points matched on either hump.

For this reason it was decided to simplify the problem of finding the correct ceiling edge by mounting the cameras more parallel. This has the effect of widening the curves in figure 2.7, thereby limiting the visible range of the cameras to the area between the tangents to the wall and ceiling in figure 2.8, so that in this range, greater depth actually does correspond to minimum disparity.

	Bearing	*Actual*	*Experimental*				
			1	*2*	*3*	*4*	*5*
1	N	13	* 52	* 26	* 26		
	W	18	16	16	16		
	S	20	* 10	17	* 10		
	E	6	6	8	6		
2	N	22	20	19	20		
	W	18	* 30	* 9	16		
	S	11	9	10	12		
	E	5	6	* 32	6		
3	N	20	* 13	19	* 10	* 52	* 40
	W	8	9	8	9	12	9
	S	13	* 62	12	12	14	12
	E	15	13	17	* 9	14	14
4	N	18	16	16	16		
	W	11	11	11	11		
	S	14	19	* 60	* 60		
	E	12	12	12	11		

Figure 3.13: The results of the roomfinder in several trials from four different locations in a room. The starred numbers indicate values which are extremely inaccurate.

3.3.4 Experimental Results

For this experiment the robot was placed in four different locations in a large room filled with furniture, clutter and people, and was provided with the value of $\phi = 33°$. The results were multiplied by the height of the ceiling for human understandability. The results of the roomfinder in several trials from the four locations rounded to the nearest foot are shown in figure 3.13. Figure 3.14 shows the four locations in the room where the experiment was performed.

3.4 Self Calibrating ϕ

For this experiment the robot moved to successive positions across the room, taking note of the ceiling edge of opposing walls. Note that for calibrating, it is

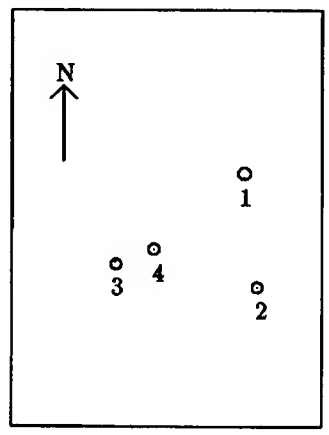


Figure 3.14: The four locations in the room from which the roomfinder was tested.

Wall 1	*Wall 2*
348	114
328	136
317	190
112	224
417	268

Figure 3.15: The y offsets of the pairs of ceiling edges in the bottom camera.

unimportant whether or not the robot is facing a wall; the technique requires only that the walls be parallel. The y offsets of the pairs of ceiling edges in the bottom camera (which is nearer to the robot's center of rotation) in pixels are shown in figure 3.15.

These points were plugged in pairwise into equation 2.8 for successive values of ϕ, as explained in the previous chapter. Since there are five curves in each group, there are ten intersection points. These are rounded off to the nearest degree in figure 3.16. The table indicates the fourth point does not intersect any of the other curves and so is a bad data point. In general, the curves are very sensitive to small errors in y, and as the ceiling edges returned are only about 75% reliable, they cause large errors in the calibration calculations as can be seen.

When the ceiling-edge pairs were hand picked and fed to the program in a

Angles of Intersection

	1	*2*	*3*	*4*
2	86			
3	46	29		
4	X	X	X	
5	26	26	26	11

Average Angle: 35.7°

Manually Measured Angle: 33°

Figure 3.16: The ten intersection points of the curves defined by the edges in figure 3.15

.

Angles of Intersection

	1	*2*	*3*	*4*
2	39			
3	37	34		
4	37	35	36	
5	37	35	36	29

Average Angle: 35.5°

Manually Measured Angle: 35°

Figure 3.17: The results of self calibration when the ceiling edge pairs were manually picked.

different run, the results were reasonably accurate (3.17). However, this is not a fair criterion by which to judge the technique.

3.5 One Last Experiment

Though the algorithm which calculates distances is not always reliable, it does suffice to do interesting tasks such as traversing the diagonal of a room. This is done by spinning in place, locating the walls (which is very reliable), picking what the robot thinks is the farthest diagonal (less reliable), turning towards it and heading there for several feet, then checking again to readjust the direction to the

chosen corner.

This experiment was repeated several times with great success. Though the robot did not always pick the furthest diagonal initially, it always did pick some corner, most of the time one which bordered one, if not both, farthest walls, and could relocate the same corner again and again after each spin. The robustness of this process is due to the fact that even when the robot got the distances wrong, thereby picking an angle that does not point directly to the destination corner, it would get another chance, and eventually did meander into the corner where it tended to want to crash into the wall, not seeing the ceiling junction once it got there.

3.6 Evaluation of Results

The results from roomfinder can be broken down into several stages, each of which can be evaluated separately — wallfinding, self calibration, and wall distance estimation.

The wallfinder worked very robustly, and is the single most successful stage of the process, failing in extremely few cases. Its success is due to the minimality of the information from which it can infer far more: namely, the fact that it needs to identify only a single wall in order to determine the perpendicular direction to the other three. This would suggest that behaviors be designed which depend as much as possible on the information from only this stage. The robot's traversing the diagonal of a room is an example of such a behavior.

Since the calibration stage is very sensitive to errors, it is more likely to benefit from a very large number of data points from which the obvious outliers can be eliminated. This suggests that the calibration technique be used not to initially calibrate ϕ, but rather to track drifts in ϕ from its initial value over time, since the time required before startup to collect enough data points for reliable initial

calibration would be great, whereas each reliable pair of opposing walls can be used to update ϕ as the robot maps.

Though the roomfinder seldom correctly ranges all four walls during the same sweep, it often correctly ranges opposing pairs of walls. If we assume that on the average one of four readings will be incorrect, then from every room sweep one of the dimensions will be correct. The incorrect values are recognizable because they will not conform to previous readings. If the robot also has the ability to recognize doors, it can also use the information that it expects to be in the same room (because it has not passed through a door) to prune incorrect readings.

Note also that certain locations in the room yielded incorrect ranges to particular walls, but moving a few feet to a new location corrected the misconception. This can also be used to prune bad data points.

If the results of the matcher are reliable, then we can determine the location of the wall-ceiling junction even in the absence of a brightness change at that point by fitting two perpendicular lines along the other points in the image, which will usually fall along a wall or ceiling, and then determining the position of their intersection.

Chapter 4

The Doorfinder

In order to connect rooms in a node map, which is the higher level goal of this work, the robot must be able to move between rooms through doors; thus, the need for a doorfinder. This chapter presents the theory behind how the doorfinder operates, and some experiments are presented.

4.1 Self Calibration and Depth Estimation

The doorfinder is based on the work described in [BFM87], the main ideas of which I will present here. Two roughly forward pointing cameras whose configuration relative to each other is unknown are mounted horizontally on the robot, and go through a calibration stage in which the robot moves forward at an unknown but constant velocity for some number of images and tracks the horizontal motion of vertical features across its field of view. Like the roomfinder, the doorfinder uses only a one dimensional strip from its field of view; however, the doorfinder uses horizontal strips to track vertical features, while the roomfinder uses vertical strips to track horizontal features. If position for a feature is plotted against time, then every feature traces a hyperbola in this plot (figure 4.1).

Using the motion of these tracked edges, the center of expansion for each

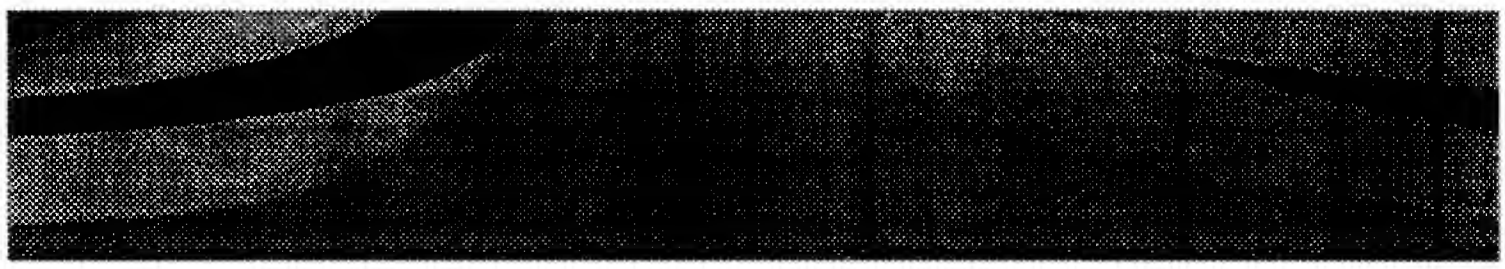


Figure 4.1: A plot of a single horizontal strip from the middle of the camera against time (y axis).

camera is deduced. The time to collision in images for any particular horizontal feature is given by the equation:

$$z = \frac{r - c}{\dot{r}}(1 + \frac{cr}{c^2 + f^2}) \tag{4.1}$$

where c is the center of expansion, f is the focal length, r is position of the observed feature in the image plane, and $\dot{r}$ is its velocity in pixels per frame. All distances are measured in pixels from the center of view in the image plane. Next, the $1D$ features are matched in the right and left images. Every matched pair of features which have comparable depth estimates (where depth was determined according to equation 4.1) is considered to be a "correct" match, and is used to calibrate the following two parameters:

$$\Lambda = \frac{n\sum(d_{1i} - d_{2i})/z_i - \sum(1/z_i)\sum(d_{1i} - d_{2i})}{n\sum(1/z_i^2) - (\sum 1/z_i)}$$

$$\Gamma = \frac{\sum(1/z_i)\sum(d_{1i} - d_{2i})/z_i - \sum(1/z_i^2)(\sum d_{1i} - d_{2i})}{n\sum(1/z_i^2) - (\sum 1/z_i)^2}$$

These two parameters are in turn used to determine depth according to the approximation equation:

$$z = \frac{\Lambda}{\Gamma + d_1 - d_2}$$

where d_1 and d_2 are the horizontal position of the same feature in the two cameras, and z corresponds roughly to linear distance provided that the tilt of both cameras

from the direction of motion is within 5° (For a more detailed explanation, please refer to [BFM87]). Note that the units of z are in images; the time per image pair must also be known in order to know the time to collision of any particular feature.

After the initial calibration stage, the centers of expansion, c, are known for each camera, as well as the parameters Λ and Γ, which will provide depth information (in terms of number of frames to collision) of features matched in the two images.

Note that in the above analysis, the data points used to calibrate the parameters Λ and Γ were used only after they had been corroborated from two independent sources, namely, time to collision estimates from the forward motion analysis from each camera. This use of redundant information helps to prune out the bad data points. It would seem as though a similar technique of redundant information cross-checking should apply to the roomfinder problem; however, in the doorfinder configuration it is only the fact that the angle between the optical axis of either camera and the direction of motion is confined to within 5° that allows for a reasonable time to collision estimate. In the roomfinder configuration, the angle of tilt is generally greater than this; so the time to collision analysis does not apply.

4.2 Implementation

The previous section described how to perform self calibration and depth estimation given two forward pointing cameras. Given this algorithm, the doorfinder is a straightforward implementation of these ideas. It continually monitors the world out of the two cameras, taking two snapshots and matching edges along the middle horizontal strip of both images to get depth information from stereo, and looking for a door in the resulting depth pattern. A door is defined as two edges

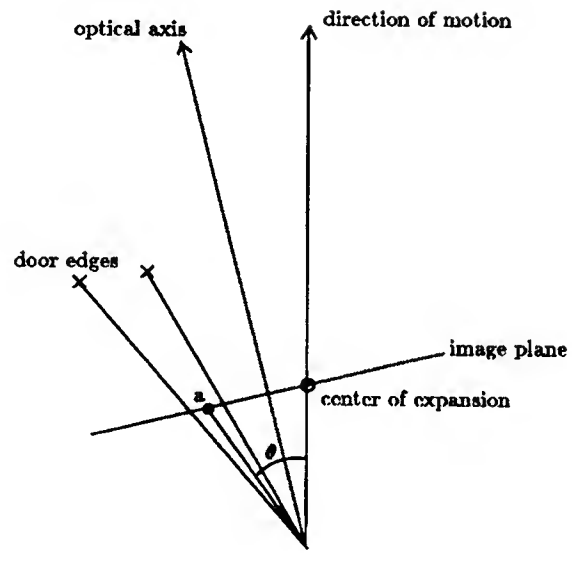


Figure 4.2: The configuration of the center of expansion with respect to a potential door in a single camera.

at roughly the same depth separated by at least one edge of greater depth. The assumption is that the ability to see an edge implies that there is a free path to it; therefore the edge separating the two edges of comparable depth implies that the two edges are traversable. When the robot finds such a pattern of edges, it turns such that the two door edges are evenly spaced around the center of expansion of both cameras, and moves towards it. The desired angle of rotation for each camera is:

$$\theta = \tan^{-1} \frac{d_1 + d_2}{2f} - \tan^{-1} \frac{C_E}{f}$$

where C_E is the center of expansion in the camera calculated from the self calibration stage, f is the focal length, and d_1 and d_2 are the horizontal coordinates of the two door edges in the image plane. Each camera yields a value for θ which is averaged together for the final rotation angle. Figure 4.2 shows the configuration for a single camera.

The hypothetical door, once chosen, is not rechecked, since both door edges move out of the image plane of the cameras almost immediately as the robot moves towards it, due to the very narrow field of view. This means that the doorfinder

at present works "ballistically"; it points towards the door from a range of over ten feet (it can't see both door edges much closer than this) and "fires" at it. This requires a fair amount of accuracy from the initial rotation angle calculation in order to successfully traverse a door.

4.3 Results

The experiments with the doorfinder were performed with two forward pointing cameras with $16mm$ lenses, each having an angular range of only about 20°. The robot performed six self calibration runs, all of which resulted in accurate values for the center of expansion, but only one of which yielded reasonable values for the calibration parameters Λ and Γ. The good run yielded the values $C_{Eleft} = 276$, $C_{Eright} = 330$ in pixels, and $\Lambda = 2633$ and $\Gamma = 83$. The values of the calibration parameters from hand calibration were $\Lambda = 2709$ and $\Gamma = 89$, which were very close to the self calibrated values. However, in the succeeding experiments the hand calibrated parameters were used, since the depth estimation is very sensitive to inaccuracies in these values.

Upon looking at the failures of the self calibration runs, it was noticed that the calibration parameters were extremely sensitive to small errors in the center of expansion estimation. Since the cameras' orientation relative to the direction of motion was constantly shifting due to unevenness in the floor and the movement of the camera mount relative to the base from jiggling, the centers of expansion would sometimes move within a single run, affecting the time to collision estimates of matched points and yielding very few data points for the calibration.

However, using the hand calibrated parameters, the robot was able to locate and maneuver its way through doors as follows: the robot was commanded to rotate, checking edges after every 1° step, until it found what it thought was a door. At this point it calculated the angle through which it would have to

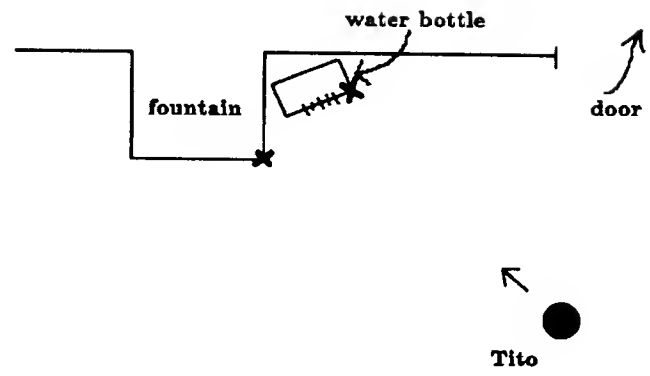


Figure 4.3: In this birds-eye view, Tito is facing towards the fountain and sees two edges at approximately the same depth, marked with x's in the drawing. However, it is confused by the similar looking edges on the water bottle, and incorrectly thinks that one of these is far away.

turn in order to line its direction of motion through the midpoint of the door, as explained in the previous section, and finally it would rotate and translate forward for as many images as required to get it just past both door edges (remember that time for the robot is measured in images, and with constant velocity this gives distance).

In general, during an angular sweep of a door, both door edges are visible in both images for about twenty 1° steps, therefore the doorfinder has several chances to find the door even if it doesn't recognize it the first few times. The more serious case is when the doorfinder finds spurious doors; this problem occurs, again, due to a matching failure. In over a dozen runs, the doorfinder found a true door over half the time during an angular sweep and successfully turned towards it and traversed it. Unfortunately, it identified spurious doors almost as often. This usually was due to finding two edges at the same depth correctly, but matching the separating edge incorrectly, thinking it was further away than it really was. It particularly liked the area near the water fountain by the open door, which happened to contain many similar looking vertical edges. The matcher would consistently be confused by these, and the robot had to be prevented from crashing into the water fountain more than once (figure 4.3).

4.4 Evaluation of Results

The doorfinder's functionality has two separate parts, self calibration, and actual doorfinding using the calibration results. In experiments with these two parts it was found that doorfinder's self calibrating stage is very sensitive to small errors in the center of expansion. This problem might be attenuated by very large numbers of data points which will be continually collected as the robot runs, since the correct values are expected to cluster around a single value, allowing outliers to be dropped. However, at the present time this stage does not work reliably.

At present, the doorfinder has successfully managed to locate and steer the robot through doors in about two thirds of the trial cases. Identifying two edges which are not door edges is not considered to be a failure unless it causes the robot to run into walls, which it has occasionally tried to do. Note, however, that in a complete system other subsumption layers would be protecting against such collisions.

Chapter 5

Mapping Issues

This chapter is a change in pace from the previous technical chapters. In it, I want to explore the issues of what a map is, what it is used for, and how maps are represented and referenced. I also make the initial assumption that human beings are the best example of map builders and users known, and are thus valid subjects to emulate when porting the problem to robots. The ideas discussed here are intended to motivate the subsequent chapter, which will discuss how the modules developed in this thesis can interact to build and use navigable maps.

The first part of the chapter will discuss issues of map representation, the next part will discuss psychological studies on humans, and finally I will talk about what principles can be used to shed some light on possible robot map representations.

5.1 What is a Map? Some Representation Issues

Before starting to talk about robot mapping, we can first put the entire analysis into the context of the higher level question of — what is a map, and what is it used for? This at first seemingly simple question can, as all simple questions, become more complicated the more one looks at it. An obvious answer is: a map

is an accurate model of the knowable world, i.e., the accurate representation of the location of all objects in the known world.

This model of the world is used in several ways —

(a) to assign the user an absolute location within the model,

(b) to know the locations of other objects or "places", as yet undefined,

(c) to use the information of (a) and (b) to perform tasks, either of manipulation (collecting soda cans, placing block A on top of block B) or of navigation (getting from where one is now to where one wants to be).

Another question to ask is, is a map a static or dynamic entity? A model of the world can be arbitrarily precise — for it to be perfect, it would have to be dynamic. If we look at our own ideas of what a map is, it is clear that we generally consider a map to be not an exact representation of the world, but rather a static entity which models only those things which are invariant. For example, a floor plan of a building includes walls and doors, but not furniture, and a map of a city includes streets, parks, and even monuments, but not a travelling circus camping in the middle of a park, even though the circus may be physically larger than other sites which might be included in the map. We have defined a map at this scale of precision due to (a) the ability to distinguish between static and dynamic objects, (b) the ability to use those objects which we consider static as reference points, and (c) the ability to react to the unpredictability of dynamic objects in real time. In other words, given our perceptual abilities, we have defined maps to be at the scale at which we can use the immovable objects represented therein as global reference points.

5.1.1 The Problem of Planning

Chapman has demonstrated that domain dependent planning is semi-decidable [Cha85a]; this means that if a plan which would achieve a particular goal exists, it

can be found by a computer given the appropriate inputs, but if it doesn't exist, the computer may never be able to discern this and would loop forever looking for the solution. This development is taken by some researchers as a justification for rejecting planning in subsequent work in robotics, and is particularly evident in the work of Connell [Con88] and Horswill [Hor88]. Much of the work done by Brooks and the MIT mobot group is concerned with exploring reactive behavior as an approach to robot control, and as an alternative to representation. Reactive behavior is on the other end of the spectrum from planning. The term refers to the concept of acting on the world according only to what stimuli are instantaneously acting on the agent. Implicit in this model is the idea that there is no state or memory associated with the agent. However, there is a well-defined middle ground which is a natural outcome of the previous discussion on mapping.

There is no reason to dismiss the ideas of planning and state because they cannot solve the comprehensive problem of action. If we limit their scope to those problems for which they are suited, they serve us well. By the same token, it is a fallacy to reject reactive behavior because its critics claim that there is a limit to the behavioral complexity that this approach can produce. Because dynamic objects' movements cannot be instantaneously modelled or predicted, the interactions between the robot and such objects cannot be planned and are ideally suited for the application of reactive behaviors. This point naturally leads to the idea of behavioral decomposition into planning and reactive procedures that together determine the actions of the robot, an idea inherent in the subsumption architecture, discussed in [BC86].

To apply this behavioral decomposition to the problem of navigation, it is clear that there will be a need for at least two kinds of behaviors:

- Planning behaviors: the problem of planning is undecidable, but if the domain is limited to a finite number of objects, the search space of all possible plans is also finite and therefore decidable. For a navigating behavior which

uses a map to determine location, the number of rooms is finite and therefore is an example of such a domain.

- Reactive behaviors: these behaviors will deal with dynamic and unpredictable objects in the world, i.e., all those things which the previous module cannot model.

The idea of behavioral decomposition into planning and reactive modules is not new, and is already inherent in the control architectures of at least one implemented system [DHK+88].

5.1.2 Performing Localization and its Inherent Problems

The above discussion renders the issue of mapping and navigation again into a deceptively simple problem; it would seem to suggest that the major problem facing navigators is the classification of all objects in the world into static and dynamic objects; once this is done the appropriate behaviors can work in coordination to perform "navigation". However, even given this classification, there is still a major source of uncertainty inherent in this analysis; the instantaneous position of the robot within its own model of the world. There are two fundamentally different techniques which are used to update the robot's actual position in its map. They can be used either separately or in conjunction. These are: (a) trajectory integration and (b) local feature identification and subsequent motion deduction. However, neither of these techniques is infallible. Trajectory integration is limited by the precision of the odometers; the resultant cumulative error is unbounded with time. That is, the robot is absolutely self-centered and lives in a solipsistic world, one which diverges from the real world as time passes. Therefore, unless technology reaches perfection, it is unwise to expect to ever be able to accurately perform localization using only this information.

The second technique, feature detection and motion deduction, has been successful by and large only in static environments due to our present incapacity to adequately identify particular objects in real world scenes; this information is a prerequisite for distinguishing dynamic from static objects. However, since this latter technique is limited only by our present information processing capabilities and not by any inherent technological limit, it is reasonable to assume that the more successful route to accurate localization lies in those methods which are based on the identifiability of places [KB87, KB88b], whether or not we can accurately deduce this information with present sensing and processing techniques.

Dean has done work in classifying different kinds of map building problems according to computational complexity bounds involved in updating and referencing them [Dea88]. Three of the relevant kinds of problems he discusses are —

- The "alert commuter" problem, in which sensors are perfect, and thus discrete locations uniquely identifiable, and every transition between locations is taken note of by the robot. The only uncertainty is in the transition function, which has an uncertainty associated with its direction of transition. The problem at hand is determining the correct ordering of locations in the map.

- The "myopic commuter" problem, in which there is uncertainty associated with both movement and sensors, so that locations are only probabilistically identifiable. For this problem, the size of the map is unbounded with time (given that no information is ever discarded), and the set of all places the robot could be is also unbounded with time.

- The "grid world" problem, in which the robot can move in any direction in an n dimensional grid, and has absolute orientation information, for instance, an accurate compass. For this problem it can be shown that building and referencing the map can be done in time polynomial to the number

of locations, and the set of possible locations after each transition is also bounded.

Given the computational complexity of these problems, it would seem as though we would do best to have a perfect sensor, so that we could uniquely identify the robot's location. Though this may be an unreasonable goal to hope for, there are ways to increase the identifiability of locations, using only the doorfinding and roomfinding sensors presented here, which will increase the chances of turning the mapping problem into an even more simple variant of the "alert commuter problem".

5.2 Intuitions Taken from Psychological Observations

The previous section raises several issues which make navigation appear to be a difficult task, yet biological creatures are able to do it, even the least developed ones, such as ants and bees [Wil71]. While some have turned to these creatures for inspiration, the fact that researchers can communicate with humans can also provide insight as to the psychological devices used to represent locations and their relationships with each other, and how these devices serve to overcome the difficulties inherent in navigating.

The question of how to define location is not a trivial one; location can be defined at different scales of relativity, none of which is absolute. For instance, if I am sitting on a particular chair, am I in the same location if I move the chair across the room? — or if I move the building which I am in to another state? — or if the earth revolves slightly around the sun as I sit motionless in my chair? — and so on. People seem to be able to define location at several scales, for example, a circus encampment relocates to different cities, yet the maintenance of a fixed internal arrangement of tents serves to create the impression among the performers

that they are in the same "place" [Rel76]. Generally, it appears that people define location at a scale commensurate with their perceptual abilities, calling a location "absolute" when in fact they mean "to the best of my ability to distinguish". This explains the difficulty that a robot (whose perceptual capabilities do not rise above the probabalistic identification of local small objects) has in localizing itself once the trash can, whose absolute location it depended on, is moved to another part of the room.

5.2.1 How Do People Do It?

Common observation is sufficient to demonstrate that humans have a very poor sense of absolute distances and angles; for example, you, the reader, cannot accurately estimate the distance to the nearest wall even though you can clearly see it, nor can you draw a good approximation of even a 30° angle without reference to a 90° one. It is then highly unlikely that people use distance and angle measurements in their mental maps, but rather use other more general principles to organize their conception of space. The manner in which humans internally represent the space they inhabit and how they locate their own position in their internal map is a fascinating subject. Although far more sophisticated than anything robots can do now, some observations taken from humans can point to a compact and useable representation for robots. Kevin Lynch, in his oft-cited work *Image of the City* [Lyn60], hypothesizes that people tend to represent cities in terms of the basic building blocks:

- paths — these lead from place to place.
- edges — linear elements, *not* paths, which serve as boundaries or barriers.
- districts — larger areas unified by some similar attribute, e.g., style.
- nodes — serve as origins and destinations of any journey; one can enter and apprehend a node in its entirety.

- landmarks — something which one cannot enter, but which is distinctive or recognizable from the outside, sometimes merely by virtue of its visibility from many locations.

Although this is an interesting breakdown into building blocks for entire cities, some of these blocks are not necessary when we scale down to the task of representing a single floor of an office building. In particular, districts are unnecessary, and edges are redundant, since they exactly coincide with walls which define rooms, which I will assume people use as nodes or "locations" within buildings.

5.2.2 Connecting Locations

Lynch suggests that in the internal mental map, locations are connected by paths which are distinctive or recognizable in some way, but leaves it to the reader to infer what exactly is meant by "distinctive". It is reasonable to assume that landmarks play a part in this distinctiveness measure. The distinctiveness of a path is sufficient to navigate between two places; if a person knows that a yellow brick road is the distinguishing characteristic of the path to Oz, he need not know the direction it leads in order to arrive there. Kuipers, following Lynch's building block model, has hypothesized a mental model which makes use of the cues received while actually on a path to direct its own continuation; this model tries to account for the phenomenon of a person being able to get to a destination, but not being able to explain how to get there unless he is actually en route [Kui83]. In some of his later work on mapping with Byun, Kuipers characterizes paths by the robot behavior utilized in order to follow them [KB87, KB88b], as does Connell [Con88].

It is clear that landmark recognition plays an important part in location identification and path following in humans; this does not necessarily imply visual ones. In the absence of visual landmarks, other unmistakable landmarks are utilized. In a study of how blind people navigate around the MIT campus [KG73], the blind

people reported that the most distinctive place for them was Lobby 7 because of the echos of the large dome, and that they use this place as a reference to get to other places. One person reported that he always goes to Lobby 7 and then makes his way to his destination from memory, using recognizable landmarks such as stairs, large rooms, or water fountains. These latter objects are recognized by sound and air pressure; the blind people described the ability to localize using smell and heat sensing as well. Subsequently they suggested that maps for blind people include such features. Many of the blind people also mentioned how they will go out of their way to avoid crossing areas devoid of clues and landmarks, such as parking lots and fields, so as not to get disoriented.

Another approach to connecting locations is by approximate distance and relative direction, much like in a traditional map. People clearly do use this information to some extent, since the directions "Turn left at the light and go for three blocks" would be meaningless unless this were true. However, this representation is limited in its usefulness, as nearly all studies of cognitive mapping have indicated that people recognize and utilize right angles very well, but have trouble comprehending angles that are not discrete multiples of 90°; when encountering such an angle, people will tend to go through all sorts of mental contortions to try to represent it as a right angle at any cost. This explains why everyone gets disoriented at the 5-sided Boston Commons [Lyn60], and why blind people avoid any intersection that is not 90° [KG73]. Therefore, it is unlikely that people use very accurate orientational information in their mental maps.

The way blind people get from place to place suggests that they know the spatial relationships between adjacent locations, but they do not abstract these spatial relationships upwards to place entire sequences of places into a global framework of spatial relationships. For instance, a blind person will use the information "turn left at the water fountain and go until you get to the stairs", but he will not realize, after walking in a loop, that he is very close to his starting point,

and will not invent shortcuts that would take advantage of the nearness of distinct locations. Sighted people do have this ability to varying extents. However, as it is a fact that blind people do navigate, it would seem that the information total ordering of spatial relationships between nodes on a map is helpful but not necessary to the manner in which people navigate.

Given that people do have a sense of relative positioning of adjacent places, though often inaccurate, and given that people can reach places whose distance and angular bearing from their present location is unknown, it is probable that they use path distinctiveness information along with distance and orientation information on a very local level, connecting only a few nodes at a time, in order to connect places in their mental maps.

5.2.3 How Distance and Direction Information is Stored

Studies on cognitive mapping in children and adults have yielded interesting hypotheses on the manner in which people mentally store locational information. In one study, people were walked around a large room which contained several distinctive objects, and were then put in a corner of the room behind a screen and were asked to point to the position of objects which they had just seen. The result was that most people seemed to believe that the large objects, which were not symmetrically located in the room, were directly across from each other along the room's axes [HMP76]. Apparently the adults tended to create symmetry where in actuality there was none, and it was suggested that they do this in order to sacrifice "absolute accuracy for the sake of cognitive economy." It was noted that this tendency to create symmetry increased with age. It was also observed that most of the children could only locate objects by seeing them, pointing in fairly random directions that had no relation to the objects' actual locations when they were not visible. These results suggest that the ability to store location information and storage efficiency develop with age.

In the same study, people were placed in a particular location in the room and were asked to imagine that they were standing in a different location in the same room. They were then asked to point to where particular objects would be, were the subject actually at the imagined location. It was noted that in their responses, people tended to make a quick decision about what direction the object was in, and proceeded to fine tune the angle of the original guess to their final answer. Pick suggested that this was an indication that people have two encodings for object locations, general direction at the top level, and then angle. It is unfortunate that this study did not result in any insights into distance encoding as well.

5.3 A New Approach to Mapping

In any case, the results of these and other studies seem to suggest that the maps that people make and use are at an extremely gross scale, rife with inaccuracies, and yet well suited to the purpose of navigation. It is my goal to apply the hypothetical principles of human navigation to robot navigation, hopefully adding some insights and robustness in the process.

The goal at hand is to build a representation and a control strategy which would enable a robot to build and utilize a map of a floor of an office building. An office environment is a much simpler place than a city, and the representation of only nodes and paths which connect them is sufficient to understand the structure of an entire floor. In the current plan, it is the rooms which serve as nodes, and I have outlined in chapter 3 a means for identifying them. The choice of nodes being defined at the scale of rooms is optimal for the following reasons:

- Rooms are clearly topologically invariant, insensitive to dynamic environments, and are the smallest topological unit for which this is so, and
- People use rooms as nodes (in the Lynchian sense), and it is desirable to have a robot's map directly correspond to those used by people for several

reasons, not the least of which is that we hope to build robots that act like people.

In Lynch's scheme, nodes are connected by paths, and in the context of an office building where the nodes are defined as rooms, the paths will be defined as the doors which connect them. There already exists a behavior in the subsumption domain suited for the task of identifying doors, discussed in the previous chapter.

If we have extremely accurate sensing capabilities and can count on the reproducibility of readings from a single location, then localization can be done using landmarks in a probabilistic fashion. One can think of the interpretation of the data from all of the sensors as a sort of landmark; hopefully the interpretation will point to only one possible locality for the robot [Mor88, KB87, KB88b]. But in the presence of dynamic objects this method will not work, and more conventional landmarks are essential for localization. However, visual landmark recognition identification is very difficult since it requires the ability to recognize 3D objects from their 2D representation, an entirely separate problem which is beyond the scope of this thesis.

Since the robot's visual processing capabilities are limited, it may be useful to artificially designate a particular room as "home" by placing in it some object that the robot can recognize unequivocally. Without any landmark recognition capabilities at all, it would be difficult if not impossible for a robot to recognize that a corridor had gone around the building and led back to the same place.

Some of the observations of human navigation presented here serve as justification for the work presented in the next chapter, which will discuss the map building and referencing strategy the robot will use given the roomfinding and doorfinding modules discussed in the previous chapters.

Chapter 6

Building and Referencing the Map

This chapter builds upon the work presented up until this point by explaining how the techniques for finding doors and rooms presented in the first few chapters might be made to work in conjunction to produce useable maps. Some of the psychological observations and computational complexity issues raised in the previous chapter are brought in to serve as justification for several of the representation choices that were made along the way.

6.1 Using the Doorfinder and Roomfinder to Determine Location Change

Let us first start with the simplifying assumption that the roomfinding module returns correct information, and see how with this alone, a node map of the room connectivity of a simple environment can be built. Other work in mapping has made use of the concept of "distinctiveness of sensory input" to individuate locations [KB88b]; the advantages of defining different rooms as distinct locations or "nodes" are:

- Location becomes a discrete, not continuous, concept. Anywhere within a single room is defined to be the same location, and one changes location only when one passes through a door. Given a mechanism which accurately recognizes doors, the problem of when one has changed location becomes easy.

- The concept of different rooms being defined as distinct locations corresponds to the manner in which people localize themselves within buildings — thus, a robot which navigates like people.

The doorfinder, which is really only a vertical line finder, is only part of the mechanism which determines location change. It is as likely to home in on two chairs standing near each other as it is to find an actual door. The way out of this bind is to utilize the roomfinder to verify the room change once the doorfinder has identified and carried the robot to a potential door.

Consider the room layout shown in figure 6.1. This is a representation of an actual sequence of offices in the MIT AI lab. Suppose the doorfinder identifies a spurious door somewhere within the room. The doorfinder then carries the robot up to the candidate door, and the roomfinder is queried about the size of the room and the robot's location within it. The robot subsequently moves through the two edges and the roomfinder is again queried. In the case of an actual room change, the robot's position and orientation within the room changes significantly; if the position of the robot with respect to the room has not changed, then the two edges through which the robot moved did not describe a door. In the former case the robot starts by facing a nearby wall and ends with a wall directly behind it facing open space. This information allows the robot to identify a room change even when the exited and entered rooms have the same dimensions.

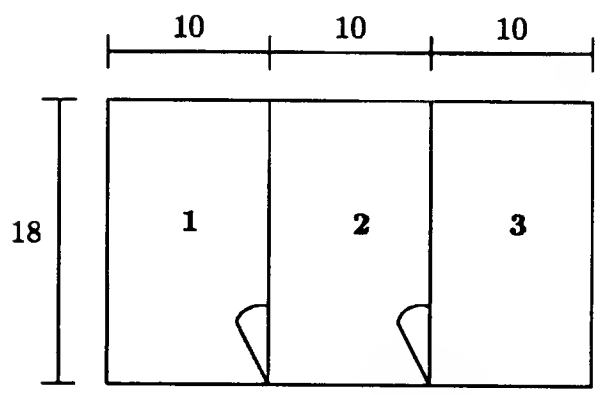


Figure 6.1: The layout of three rooms in the AI lab.

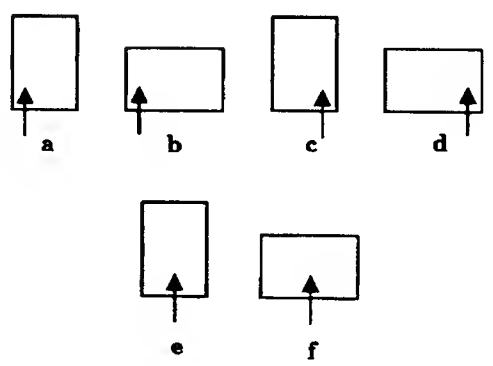


Figure 6.2: Different positions of doors with respect to the room.

6.2 Building the Map

In building and using a map, the robot must be able to distinguish between two rooms that may look the same even when they are not. In the current conception, a room is distinctive not only by its dimensions, but also by the paths taken to get there, where the paths in this case are the doors leading in and out of it.

Let us divide all doors into three gross categories: those that are positioned toward the left side of the containing wall when one is facing into the room with the door behind one, those which are towards the right, and those which are in the middle. This allows the roomfinder and doorfinder working in conjunction to distinguish between 6 non-isomorphic room/door configurations for a non-square rooms, and 3 room/door configurations for square rooms (see figure 6.2). Every time a door is traversed, two nodes connected by a door/door path are created. For example, upon passing from room 1 to room 2 in figure 6.2, the mapping module would use the information provided by the roomfinder and doorfinder to construct the node pair pictured in the top part of figure 6.3.

The proposed mapping algorithm works as follows: upon startup,

- Initialize the map by entering a node with the current room's dimensions.
- For every door found, construct a node pair as described. If the node pair is consistent with one already leading off from the current room, then we are in previously charted territory and have just entered into a known room. If not, we have just found a new door and room. Enter this new node pair into the connectivity map.

This algorithm creates the connectivity map pictured in the bottom part of figure 6.3.

This statement of the problem corresponds almost directly to the alert commuter problem, since for that problem, all locations are distinguishable, the 1-1 adjacencies are easily available, and the problem is the directional ordering of

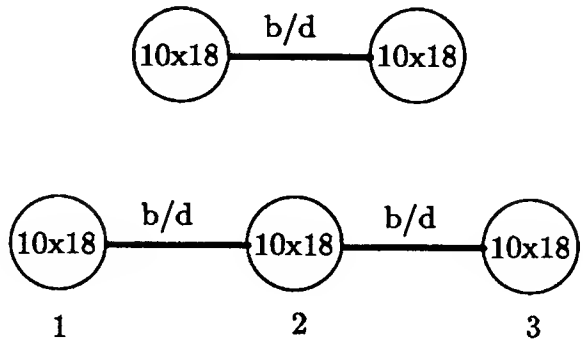


Figure 6.3: The connectivity map built from the configuration of rooms in figure 6.1.

these adjacencies. In this problem, though location is not uniquely distinguishable (since there may be several rooms with the same dimensions and distinguishing doors), the possible current location is completely constrained by where the robot was one step before. Furthermore, we are interested in the adjacency graph, not in the total ordering of the nodes in the graph. The disregard for spatial ordering here is inspired by the way in which people are observed to store the relationships between their map nodes, and significantly simplifies the representation. Thus, the problem is already solved.

However, the mapping algorithm as it stands cannot handle any loops in the environment — for example, the same two rooms connected by two different doors would be considered to be different node pairs and would cause an incorrect doubling of the connectivity map. It also incorrectly handles the case where a single room contains two doors which look exactly the same from both sides of the doors; in this case the two doors would be identified as one and the same, and would cause an incorrect folding over of the map. Both cases are shown in figure 6.4. However, if the environment does not contain loops or isomorphic node pairs, then absolutely no odometry information or trajectory integration is needed to successfully build the node map.

If the environment does contain loops or isomorphic node pairs, then a simple on-board compass is sufficient to identify the problematic cases. If we do have a

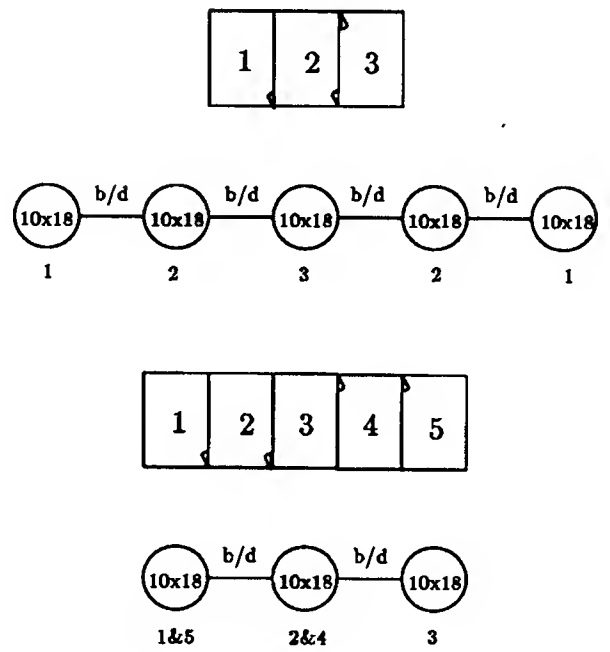


Figure 6.4: The connectivity map built from problematic situations. In the top figure there is a loop, i.e., two paths connecting the same two rooms. In the bottom figure there are two doors in the same room which are indistinguishable.

compass, then there is another four-valued distinguishing characteristic of doors, bringing the number of non-isomorphic room/door configurations for a non-square rooms to $6 * 4 = 24$, and half that number for square rooms. This is enough information to correctly disambiguate the similar looking doors in figure 6.4, and most other cases as well.

In addition, we will allow the robot to recognize a separate entity called "corridors" by reference to a predefined aspect ratio of length to width, and when the robot enters one of these, the path that it chooses will have either a 90° or a 270° angle associated with it. This directional information will be necessary in order to correctly identify adjacent geometrically identical rooms coming off a long corridor, even if the starting room is known. However, introducing this concept presents a problem, since most rooms off the corridor are likely to be of the same dimensions, introducing isomorphic node pairs leading off the main corridor node in the graph. Distinguishing between these nodes will most likely require a notion of node ordering, which I have not discussed, and landmark recognition, not likely

to be implemented in the foreseeable future.

6.3 Using the Map

Once the map has been built, how can the robot use it to navigate? There is a problem with startup, if the robot already has the adjacency map and is required to deduce where it is, given no information of previous movement. Depending on the similarity of different parts of the adjacency graph, the robot may or may not be able to deduce its location. If the graph is non-trivially auto-isomorphic, the robot can never narrow down the set of its possible locations to a single location. If it is not auto-isomorphic, then it is possible for it to localize itself given that the robot has enough memory to remember all the locations it has ever been. For instance, imagine a graph consisting of only two distinguishable room types, A and B, which are connected in a circular list of 99 A's, 1 B, then 100 A's and 1 B. Though this graph is not auto-isomorphic, if the robot could only remember at most 99 steps, it could not distinguish between the two sections of the graph.

Given that the robot knows what room it is in, then it cannot get lost by transitioning from room to room in its adjacency graph. It can also navigate between any two given rooms in a directed fashion by using the information in the graph as follows: first, it finds a node path connecting the two nodes in question. Since the edges between nodes contain information about the position of the connecting doors with respect to the connected rooms, the robot can use this information to constrain the position of the desired door in the room to at most 2 for rectangular rooms and 4 for square rooms; with directional information (NSEW per door, as previously explained) and a compass, this reduces down to only 1 possible door location for both rectangular and square rooms. We have already seen that the robot can wake up in a room, pick a particular diagonal and head towards it; in this case, it would pick where it thought the door should be,

head towards it, and when the robot gets close enough for the door to be visible the doorfinder would fire and bring the robot through the door, where the room traversing process would start again, until the desired room was reached.

6.4 Foreseeable Problems

The previous analysis of the feasibility of building and referencing a simple adjacency map to navigate was done under the assumption that the roomfinder is 100% accurate. The problem is complicated considerably by the fact that it is far less reliable than this, and brings the real problem closer to the "myopic commuter" problem than the "alert commuter" problem. Only implementing these ideas will make evident how close we can come to the ideal.

One factor which prevents the extensive testing of these ideas is the speed at which the roomfinder runs, which is about 8 minutes per room scan. This is expected to change when the processing is moved onto a digital signal processing board which has been able to perform single line convolutions and matching at a speed of 400 per second in tests, bringing the time to process a composite room image to about one second.

Chapter 7

Conclusion

This chapter will discuss briefly the shortcomings of the methods presented in this thesis, and will try to suggest areas of improvement. Finally, an overview of other approaches to similar problems will be presented, and how the work presented here tried to address the issues raised in these related attempts to solve the difficult problem of robot navigation.

7.0.1 Evaluation of Results and Future Directions of Research

The experiments carried out for this work were accurate not much more than half the time. In all of the experiments performed, it was the matching stage that proved to be the major cause of failure. Since all of the modules discussed depend on accurate results from this step, a great improvement in overall reliability would result from improving the reliability of this single stage. However, any system which is dependent on 100% reliability from the output of any other procedure is doomed to failure. In the absence of 100% reliability, the overall system should still behave in a reasonable way.

One factor detracting from the overall robustness is that the system does not try solve the problem of sensor noise at all, nor is there any form of redundant

sensing. This has clearly affected the results of all the modules. A promising direction would appear to be building a framework which would address both these issues. Moravec [Mor88] has done work in the area of probability integration using certainty grids, and others have used redundant sensing and Kalman filtering to update internal world models and correct for drift in odometry [Cro89] and to integrate data from different sensors [DW87].

However, despite the less than perfect outputs from the modules, the robot was able to perform a simple task which did not depend on the absolute reliability of the modules, namely, recognizing corners and traversing a room along its diagonal. The approach presented here tries to tackle the problem of topological mapping in a way which might be characterized as more "qualitative" than "quantitative", and as such, it does not fail more spectacularly than any other project of its kind to date.

The three modules which I have discussed, the doorfinder, the roomfinder, and the mapper, are planned to be implemented with Brooks' subsumption architecture, though they are suitable for integration into any control model involving independent agents. The mapping module is planned to be a passive module, merely an "observer" of the path taken by the robot due to other causes — random wandering, object tracking, and so on. No attempt is made by the module to actually direct the robot to unknown territory, although in the future it will be able to issue commands to the motor control, causing the robot to actually explore its environment in a directed fashion.

7.1 Related Work

Robot navigation and mapping has been a much worked on area in the past and still remains elusively difficult, given an unstructured environment. Following is an overview of some of the important work done in the field from its inception,

and I try to pinpoint ways in which the approaches succeeded and/or fell short of their goals, and what conclusions can be drawn from them.

7.1.1 Obstacle Avoidance

Moravec in 1980 attacked the problem of navigation by concentrating on visual obstacle avoidance [Mor80]. After an initial stereo calibration stage, his robot was able to plan a path to a given destination (where the destination was given as displacement relative to the robot's starting position) through a cluttered environment, picking its way through objects which it visually identified and entered into an internal grid map after every 1 meter iteration of its basic observe/plan/move loop. Stereo matching was performed by picking interesting points in one image and finding their correspondence in the other. As a result, objects which did not give rise to interesting points were not seen and sometimes the robot would plan a path through them. The robot would deduce its own motion from the movement of the objects around it, avoiding the problem of cumulative error. A major problem for the robot was that the execution time of the main loop was approximately 15 minutes, which translated into a speed of 3–5 meters per hour. At this speed, the rover cannot be thought of as performing "real-time" obstacle avoidance. However, the approach taken for stereo matching was quite robust, due to the redundancy of information given by stereo matching between nine, not two, images. This gave rise to $\binom{9}{2} = 36$ data points per feature, which tend to cluster around the correct depth value even given several incorrect matches. The success of this approach supports the argument that the best hope for correct interpretation of real world data lies in the direction of redundant information processing techniques.

Another project in visual navigation, Mobi at Stanford, was able to move down a hallway using stereo vision by matching vertical edges between successive

images, and, assuming that the vertical edges lay along walls, built a model of the corridor [TK87]. It was also able to navigate through free space by assuming that the triangle formed by the two cameras and a correct match was free, and moving through this triangle.

7.1.2 Map Based Navigation

Other approaches have concentrated on localization of the robot by correlating information obtained through sensors to a map provided to the robot for this purpose — this is appropriately called "map based" navigation. Here, the robot is required to relate its position to an absolute reference frame outside itself, and therefore has a more sophisticated sense of place than the Moravec rover. In order for map-based navigation methods to work correctly in the presence of unforeseen obstacles, the robot must be able to perceive obstacles in real time as well as perform localization, and must be able to combine long term and short term goals into a unified plan (for instance, get to the end of a hall while going around a person coming from the opposite direction).

Hughes has successfully demonstrated the ability to perform map based cross country navigation with the ALV, while avoiding obstacles not present in the map. It uses a laser range scanner to build a local cartesian elevation map and fuses the maps from successive locations to deduce the relative change of position in all six degrees of freedom. Two competing "meta" behaviors, a map based planner and a reflexive planner, together determined the final trajectory of the vehicle.

Gilbreath and Everett [GE88] describe a surveillance robot which addresses the problem of navigation in a dynamic office environment. It has an initial static environment map, and plans a path using this. During execution it uses ultrasonic range data to update a secondary local grid of occupied locations, and may alter its path to avoid unexpected obstacles whose existence seems too certain to ignore. Upon completion of a path trajectory the secondary map is forgotten, as it is

assumed that it contained only dynamic objects. Localization is done by dead reckoning.

7.1.3 Mapping the Environment

Chatila and Laumond [Cha85b] [CL85] have worked on the problem of building a map from sensor information by trying to fit together polygons of observed free space into a consistent two dimensional map. This map consists of three levels — a geometric level, which is concerned with actual $2D$ relationships between all objects encountered (which for simplicity are represented as polygons), a topological level, which groups the free space into rooms, and a semantic level, in which certain spaces are treated as paths to other spaces, such as corridors or doors. Their robot, Hilare, uses laser range scanners to build the lowest level map. To fuse information taken from different locations it uses an approach, similar to Brooks [Bro85], of associating position uncertainties with every sensor reading, and when identifying a previously seen location, projecting the small uncertainty bounds associated with the current reading backwards to previous steps, thereby tightening the bounds on their associated uncertainties such that the map coalesces into a self-consistent model. The graph of free space resulting from this stage is then heuristically grouped into connected components and topologically labelled. Using this method, Hilare is reported to have successfully mapped small regions consisting of several rooms. However, the approach is engineered to work in a static environment and it is not clear how to extend this method to cope with a dynamic one.

Bolles, Baker and Marimont have demonstrated an algorithm to construct a 3D map of free space visually by using a single moving camera which maintains a fixed orientation with respect to the direction of motion [BBM87]. The successive images from a camera moving at constant velocity sweep out an image cube consisting of a series of epipolar planes, where epipolar planes are defined by the

vector describing the motion of the camera's optical center and a feature in the world, analogous to the stereo case. The motion of a feature across a series of epipolar planes gives the distance of the feature from the camera, and using the fact that if point A is visible to a camera moving from point X to Y, then the triangle AXY describes free space, a $3D$ polygonal analysis of the observed scene can by inferred.

Ayache and Faugeras have worked on another attempt to infer the $3D$ structure of a scene using stereo vision. This approach uses the matching of line segments in the two images, using an epipolar analysis with 3 cameras instead of 2 in order to disambiguate matches in both the vertical and horizontal directions [Fau88, AF88]. At present the mobility of the robot is utilized mainly to collect successive sequences of images; the results of actual map building and navigation using this information have not been reported on.

7.2 Contribution of this Thesis

As can be seen, most work in mapping has been concerned with building an accurate 2 or 3D map of the environment, where objects are represented as polygons in order to simplify path planning. This approach generally requires some amount of trajectory integration, and cumulative errors drastically affect the accuracy of the map. In addition, obsessive concern for precise geometric modelling generally requires a static unchanging environment; changes in the actual world, such as moving a chair to the middle of a room, will confuse a robot which happens to be using the previous position of the chair to do localization.

The approach I have taken to the problem differs fundamentally from those presented here and has the following advantages:

- In indoor environments the natural building blocks are rooms, and I have suggested a method of map building and position referencing using these

blocks as the smallest topological unit, much as people do.

- The sensors developed for this purpose are matched exactly to the map-building task. In addition, they use passive sensing and require minimal calibration and processing, and are insensitive to dynamic changes in the environment.

- No attempt is made to acquire information unnecessary to the task at hand. The $3D$ structure of a room being traversed in order to get to the room three doors down is unimportant; however, the gross structure of the room along with the location of the doors is.

- The graph representation contains information sufficient to perform navigation between rooms, but not within a single room. Navigation inside rooms should be done on the fly, since the location of objects within them are likely to change — thus why waste space remembering them?

What I have tried to do with the work in this thesis is to try to break out of the mold defined by previous work by looking at the way people navigate, and trying to apply some of the observations to robots.

Bibliography

[AF88] Nicholas Ayache and Olivier D. Faugeras. Maintaining Representations of the Environment of a Mobile Robot. Technical Report 789, INRIA-Rocquencourt, February 1988.

[BBM87] R.C. Bolles, H.H. Baker, and D.H. Marimont. Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion. *International Journal of Computer Vision*, 1(1):7–55, 1987.

[BC86] Rodney A Brooks and Jonathan H. Connell. Asynchronous Distributed Control System for a Mobile Robot. In *Proceedings SPIE, Conference on Mobile Robots*, Cambridge, Mass., 1986.

[BFM87] R.A. Brooks, A.M. Flynn, and T. Marill. Self Calibration of Motion and Stereo Vision for Mobile Robot Navigation. Technical Report AIM-984, MIT Artificial Intelligence Laboratory, August 1987.

[BKT86] Kenneth R. Boff, Lloyd Kaufman, and James P. Thomas, editors. *Handbook of Perception and Human Performance*, volume I: Sensory Processes and Perception, chapter 23, pages 23–14–23–21. John Wiley and Sons, 1986.

[Bro85] Rodney A. Brooks. Visual Map Making for a Mobile Robot. In *IEEE International Conference on Robotics and Automation*, pages 824–829, St. Louis, MO., March 1985.

[Bro86] Rodney A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[Cha85a] David Chapman. Planning for conjunctive goals. Technical Report 802, MIT Artificial Intelligence Laboratory, January 1985.

[Cha85b] Raja Chatila. Mobile Robot Navigation: Space Modeling and Decisional Processes. In M. Brady and R. Paul, editors, *Third International Symposium on Robotics Research*, pages 373–378. MIT Press, Cambridge, MA., 1985.

[CL85] R. Chatila and J.P. Laumond. Position Referencing and Consistent World Modeling for Mobile Robots. In *IEEE International Conference on Robotics and Automation*, pages 138–145, St. Louis, MO., March 1985.

[Con88] Jonathan Connell. Navigation by Path Remembering. In *Proceedings of the SPIE, Conference on Mobile Robots*, 1988.

[Cro89] J. L. Crowley. World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging. *IEEE International Conference on Robotics and Automation*, 1989.

[Dea88] Thomas Dean. On the Complexity of Integrating Spatial Measurements. In *Proceedings of the SPIE, Conference on Advances in Intelligent Robotic Systems*, Cambridge, November 1988.

[DHK+88] M. Daily, J. Harris, D. Keirsey, K. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous Cross-Country Navigation with the ALV. In *IEEE International Conference on Robotics and Automation*, pages 718–726, Philadelphia, PA., 1988.

[DJKB87] Ernst Triendl David J. Kriegman and Thomas O. Binford. A Mobile Robot: Sensing, Planning and Locomotion. In *IEEE International Conference on Robotics and Automation*, pages 402–408, Raleigh, N.C., 1987.

[Dru87] Michael Drumheller. Mobile Robot Localization Using Sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(2):325–332, March 1987.

[dSV87] Arnaud R. de Saint Vincent. Visual Navigation for a Mobile Robot: Building a Map of the Occupied Space from Sparse 3D Stereo Data. In *IEEE International Conference on Robotics and Automation*, pages 1429–1435, Raleigh, N.C., 1987.

[DW87] Hugh Durrante-Whyte. Consistent Integration and Propogation of Disparate Sensor Observations. *International Journal of Robotics Research*, Spring 1987.

[Fau88] Olivier D. Faugeras. A Few Steps towards Artificial 3D Vision. Technical Report 790, INRIA-Rocquencourt, February 1988.

[GE88] Gary Gilbreath and H.R. Everett. Path Planning and Collision Avoidance for an Indoor Security Robot. In *Proceedings of the SPIE, Conference on Mobile Robots*, 1988.

[Gri85] Eric L. Grimson. Computational Experiments with a Feature Based Stereo Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):17–34, January 1985.

[HMP76] Douglas A. Hardwick, Curtis W. McIntyre, and Herbert L. Pick. The Content and Manipulation of Cognitive Maps in Children and Adults. *Monographs of the Society for Research in Child Development*, 41(3, Serial No. 166), 1976.

[Hor88] Ian D. Horswill. Reactive Navigation for Mobile Robots. Master's thesis, MIT AI Laboratory, May 1988.

[KB87] Benjamin J. Kuipers and Y.T. Byun. A Qualitative Approach to Robot Exploration and Map-Learning. In *Workship on Spatial Reasoning and Multi-Sensor Fusion*, 1987.

[KB88a] David J. Kriegman and Thomas O. Binford. Generic Models for Robot Navigation. In *IEEE International Conference on Robotics and Automation*, pages 746–751, Philadelphia, PA., 1988.

[KB88b] Benjamin J. Kuipers and Yung-Tai Byun. A Robust, Qualitative Method for Robot Spatial Learning. In *Proc. AAAI*, pages 774–779, 1988.

[KG73] Ann Middleton Kidwell and Peter Swartz Greer. *sites, perception and the nonvisual experience: designing and manufacturing mobility maps.* American Foundation for the Blind, Inc., New York, 1973.

[Kui83] Benjamin Kuipers. Representations of Large-Scale Space: Sensorimotor Knowledge. Technical report, Tufts University, 1983.

[Lau84] Jean-Paul Laumond. *Utilisation des Graphes Planaires pour l'Apprentissage de la Structure de l'Espace d'Evolution d'un Robot Mobile.* PhD thesis, Universite Paul Sabatier du Toulouse, March 1984.

[Lau86] J.P. Laumond. A Learning System for the Understanding of a Mobile Robot Environment. In *European Working Session on Learning*, Orsay, France, February 1986.

[Lyn60] Kevin Lynch. *The Image of the City.* MIT Press, Cambridge, Mass., 1960.

[Mor77] Hans P. Moravec. Towards Automatic Visual Obstacle Avoidance. In *Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA., 1977.

[Mor80] Hans P. Moravec. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. Technical Report CMU-RI-TR-3, CMU Robotics Institute, September 1980.

[Mor88] Hans P. Moravec. Sensor Fusion in Certainty Grids for Mobile Robots. *AI Magazine*, Summer 1988.

[NB88] Hatem Nasr and Bir Bhanu. Landmark Recognition for Autonomous Mobile Robots. In *IEEE International Conference on Robotics and Automation*, pages 1218–1223, Philadelphia, PA., 1988.

[OK85] Yuichi Ohta and Takeo Kanade. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(2):139–154, March 1985.

[Rel76] D. Relph. *Place and Placelessness.* Pion Limited, 207 Brondesbury Park, London NW2 5JN, 1976.

[Sar89a] K. B. Sarachik. Characterising an Indoor Environment with a Mobile Robot and Uncalibrated Stereo. *IEEE International Conference on Robotics and Automation*, 1989.

[Sar89b] K. B. Sarachik. Simple Map Building for a Mobile Robot with Uncalibrated Stereo. *Working Notes of the AAAI Spring Symposium Series*, March 1989.

[sha84] Shakey the robot. Technical Report 323, SRI International, April 1984.